PROJETO FINAL DE GRADUAÇÃO 2

SISCOM: Sistema Comitê de Classificadores para Reconhecimento e Classificação de Emoções Humanas

Mateus Leite Pedrosa

Ricardo Dias Macedo

Brasília, Julho de 2023

UNIVERSIDADE DE BRASÍLIA

DEPARTAMENTO DE ENGENHARIA DE REDES DE COMUNICAÇÃO

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

PROJETO FINAL DE GRADUAÇÃO 2

SISCOM: Sistema Comitê de Classificadores para Reconhecimento e Classificação de Emoções Humanas

Mateus Leite Pedrosa

Ricardo Dias Macedo

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Dr. Vinícius Pereira Gonçalves, _____

ENE/UNB
Orientador

Prof. Dr. Geraldo Pereira Rocha Filho, _____

CIC/UFBA
Examinador

Prof. Dra. Mariana Cabral Falqueiro, _____

ENE/UNB
Examinador

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original".

Einstein, A.

Agradecimentos

Gostaríamos de expressar nossa profunda gratidão a todas as pessoas que estiveram ao nosso lado durante esta jornada de dissertação. Primeiramente, queremos agradecer aos nossos queridos familiares, cujo apoio incondicional foi a força motriz por trás de nossa dedicação. Suas palavras de encorajamento e compreensão nos momentos de desafio foram inestimáveis e nos motivaram a seguir em frente.

Aos nossos amigos, somos imensamente gratos por terem sido pilares de apoio ao longo dessa caminhada acadêmica. Suas presenças e sorrisos trouxeram leveza aos dias e tornaram a jornada mais significativa. Cada conversa e momento compartilhado contribuíram para nosso crescimento pessoal e profissional.

Não podemos deixar de mencionar nosso respeito e gratidão aos nossos professores, cujo conhecimento e orientação moldaram nosso pensamento crítico e nos proporcionaram um aprendizado transformador.

Por fim, agradecemos aos nossos orientadores, cujo suporte foi fundamental para o sucesso desta dissertação. Suas orientações nos guiaram na direção certa, incentivando-nos a buscar a excelência em cada aspecto do trabalho. A confiança que depositaram em nós foi um incentivo crucial para superarmos os desafios e chegarmos a esta etapa tão significativa de nossa jornada acadêmica. Somos profundamente gratos a todos que estiveram ao nosso lado, tornando este momento possível.

Mateus Leite Pedrosa

Ricardo Dias Macedo

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Com a disparada dos números de casos de transtornos mentais como depressão e ansiedade dos últimos anos, é possível utilizar os avanços tecnológicos na área de Inteligência Artificial em busca de reconhecer e reduzir tantos dados alarmantes. Esse trabalho visa desenvolver um Sistema Comitê de Classificadores, denominado SISCOM, para o reconhecimento e classificação de emoções humanas a partir de imagens digitais utilizando técnicas de Aprendizado de Máquina Profundo, Algoritmos de CNNs e Sistemas Múltiplos Classificadores.

Palavras-chave: emoções humanas, aprendizado de máquina, CNN, MCS.

ABSTRACT

With the increse numbers of cases of mental disorders such as depression and anxiety in recent years, it is possible to use technological advances in the field of Artificial Intelligence in order to recognize and reduce so much alarming data. This work aims to develop a Classifier Committee System, called SISCOM, for the recognition and classification of human emotions from digital images using Deep Machine Learning techniques, CNNs Algorithms and Multiple Classifier Systems.

Keywords: human emotion, machine learning, CNN, MCS.

Capítulo 1

Introdução

1.1 Motivação

Saúde mental tornou-se um tópico de extrema relevância ao longo dos últimos anos, após a pandemia de COVID-19, houve um aumento significativo nos casos de doenças mentais em todo o mundo. Diversos estudos e relatórios têm documentado o impacto negativo da pandemia na saúde mental das pessoas.

Segundo a maior revisão mundial sobre saúde mental desde a virada do século divulgado pela Organização Mundial da Saúde (OMS), em 2019, quase um bilhão de pessoas incluindo 14% dos adolescentes do mundo viviam com algum tipo de transtorno mental. O suicídio foi responsável por mais de uma em cada 100 mortes e 58% dos suicídios ocorreram antes dos 50 anos de idade. Os transtornos mentais são a principal causa de incapacidade, causando um em cada seis anos vividos com incapacidade (30).

Esses dados alarmantes ressaltam a importância temática das emoções humanas e como a compreensão das mesmas é algo fundamental para todas as nossas relações, ressaltando o ponto de que saúde mental deve ser uma prioridade para mitigar os impactos negativos já observados na sociedade.

Dessa forma, fica evidente a relevância do estudo e análise das emoções básicas humanas. Define-se a emoção como: *"Um complexo estado de sentimentos, com componentes somáticos, psíquicos e comportamentais, relacionados ao afeto e ao humor"* (8).

Surgem diversas proposições de como alinhar a tecnologia atual como uma forma de apoio a identificação, combate e prevenção de casos como citados de transtornos mentais. Uma delas é o reconhecimento de emoções humanas partir de imagens digitais utilizando técnicas de Visão Computacional, uma área da ciência da computação que se concentra no desenvolvimento de sistemas capazes de interpretar e compreender informações visuais presentes em imagens e vídeos podendo extrair conhecimento útil a partir desses dados visuais (7), uma tarefa extremamente complexa e desafiadora.

A utilização de técnicas de Inteligência Artificial (IA) para reconhecimento de emoções humanas

não é uma novidade, entretanto devido ao alto custo computacional que algoritmos tradicionais empregam dificultavam o desenvolvimento e pesquisa desse mesmo tópico. Entretanto, devido aos avanços significativos na área de Aprendizado de Máquina (AM) outros algoritmos mais robustos e eficientes surgiram como os algoritmos de Redes Neurais Convolucionais (CNN).

Essas técnicas permitem que computadores analisem as características faciais para identificar e interpretar diferentes expressões emocionais, ou seja, a combinação de VC com o reconhecimento de expressões faciais utilizando técnicas de AM oferecem um potencial significativo para compreender e interpretar as emoções humanas (17).

Visando uma melhoria no desempenho dos modelos, é comum na literatura o uso de Sistemas de Múltiplos Classificadores (MCS), que vão além das técnicas de algoritmos classificadores individuais. Esses sistemas, também conhecidos como *"Sistemas Comitês"*, partem do princípio de que a combinação de diversas redes treinadas de forma independente podem potencializar consideravelmente a capacidade de generalização da rede (4).

O presente trabalho busca realizar o processo de reconhecimento e classificação de emoções humanas básicas a partir de imagens digitais gerando um aumento da precisão e eficácia, por meio de experimentos realizados em uma base de dados consolidada com os seguintes objetivos da pesquisa: (1) determinar os melhores modelos de classificadores individuais realizando o processo de fine-tuning da rede proposta, utilizando os algoritmos de CNNs e estabelecendo as melhores arquitetura e metaparâmetros (2) desenvolver um sistema comitê utilizando os classificadores individuais com as melhores performances observadas (3) analisar a performance dos classificadores individuais e sistema comitê desenvolvidos (4) determinar qual a melhor solução para o processo de reconhecimento e classificação de emoções humanas básicas a partir de imagens digitais.

1.2 Objetivos

1.2.1 Objetivo Geral

Esse trabalho final de conclusão de curso tem como objetivo principal o desenvolvimento de um Sistema Comitê de Classificadores, denominado SISCOM, para o reconhecimento e classificação de emoções humanas a partir de imagens digitais. Realizando a implementação de modelos de Classificadores Individuais obtidos de um processo de fine-tuning utilizando algoritmos de CNNs, visando um aumento na performance e eficácia da rede proposta para a problematização específica abordada desse trabalho.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, foram definidos os seguintes objetivos específicos:

- Consolidar um dataset gerado por múltiplas bases de dados para a tarefa de classificação de emoções humanas básicas;

- Realizar o processo de fine-tuning da rede proposta utilizando algoritmos de CNNs para determinar a melhor arquitetura e metaparâmetros para a tarefa de classificação de emoções humanas básicas a partir de imagens digitiais;
- Gerar os Classificadores Individuais a partir dos melhores desempenhos observados do processo anterior de fine-tuning dos modelos de CNNs;
- Desenvolver um Sistema Comitê com os Classificadores Individuais gerados com o objetivo de aumentar as métricas de performance da rede;
- Determinar o melhor número de Classificadores Individuais integrados para composição do Sistema Comitê;
- Analisar qual a melhor solução em relação as métricas de performance e estatísticas geradas, além do custo computacional para o processo de reconhecimento e classificação de emoções humanas básicas a partir de imagens digitais;

1.3 Organização do trabalho

O trabalho é dividido em cinco partes principais: referencial teórico, trabalhos relacionados, metodologia e implementação, resultado e conclusões.

No referencial teórico será exposto a fundamentação teórica dos assuntos que serão importantes para a o tema escolhido. Nessa primeira parte serão abordados os principais tópicos, cruciais para compreensão e continuidade na sequência dos capítulos desse trabalho.

Na metodologia e implementações será exposto as estratégias utilizadas para a resolução do problema de reconhecimento e classificação de emoções humanas a partir de imagens digitais utilizando técnicas de Aprendizado de Máquina Profundo.

Nos resultados será documentado os experimentos realizados, levando em consideração algumas métricas de avaliação em relação as execuções dos modelos desenvolvidos.

Por fim, na conclusão será realizado uma análise de todo o trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo, serão discutidos os aspectos teóricos fundamentais relacionados à classificação de emoções em imagens digitais por meio do uso de um comitê de classificadores utilizando técnicas de Aprendizado Profundo.

2.1 Emoções Humanas

As emoções humanas são resultados complexos e fascinantes das interações intricadas entre reações químicas e neurais no cérebro. Elas se manifestam por meio de mecanismos neurofisiológicos, refletindo um vasto espectro de respostas emocionais que podem ser tanto negativas quanto positivas, abrangendo desde sentimentos perturbadores até experiências gratificantes (13).

A ativação do processo emocional ocorre em resposta a estímulos ou eventos, sejam eles de origem externa ou interna. Esses gatilhos desencadeiam uma complexa rede de processos cognitivos que envolvem a avaliação e a interpretação do significado emocional da situação. Essa avaliação cognitiva desempenha um papel crucial na determinação de como respondemos emocionalmente e como expressamos ou modificamos nosso comportamento em função dessas emoções.

De acordo com Damásio (9), as emoções são elementos fundamentais em nossas ações, ideias e tomadas de decisão. Elas não apenas complementam as bases racionais, mas são igualmente indispensáveis. Sem o envolvimento das emoções, os humanos seriam meras máquinas lógicas, destituídas de uma dimensão afetiva que dá cor e significado à nossa existência. No entanto, também não são apenas feixes de emoções descontroladas. São seres complexos que integram tanto a razão quanto a emoção em nossa experiência humana, em uma interação contínua e dinâmica.

O reconhecimento de emoções tem sido objeto de estudo há bastante tempo, com o pesquisador Charles Darwin explorando essa área desde 1872. Em sua obra (10), Darwin investigou os estados emocionais, encontrando semelhanças nos movimentos responsáveis pela inferência de emoções e analisando o comportamento dos músculos faciais. Além disso, ele examinou como os seres humanos conseguem expressar suas emoções. Pesquisadores posteriores, como Ekman (15), aprofundaram as ideias de Darwin e chegaram à conclusão de que existem expressões faciais básicas universais

(aversão, felicidade, medo, neutro, raiva, surpresa e tristeza), sugerindo que os mesmos movimentos musculares estão envolvidos na formação dessas expressões faciais.

A partir dessas emoções básicas, todas as outras categorias emocionais são construídas por meio de combinações. Além disso, um estado neutro também é considerado, uma vez que pode servir como ponto de referência para a detecção dos estados emocionais. Dessa forma, a compreensão das emoções humanas vai além das expressões faciais universais, abrangendo uma ampla variedade de nuances emocionais que são formadas por meio de combinações e contextos específicos.

2.2 Visão Computacional

A visão computacional é uma área da ciência da computação que tem como objetivo desenvolver algoritmos e técnicas para que os computadores possam entender, interpretar e extrair informações de imagens e vídeos (5). Está intimamente ligada à Inteligência Artificial (IA) e tem aplicações nos mais diversos setores da sociedade, como por exemplo na medicina, segurança, automação industrial, robótica entre outros.

Um dos principais desafios da visão computacional é fazer com que os computadores consigam "enxergar"e compreender o conteúdo visual da mesma forma que os humanos. Dessa forma, os algoritmos desenvolvidos são voltados para realizar tarefas como detecção e reconhecimento de objetos, rastreamento de movimento, reconstrução 3D etc (5). Existem várias abordagens e técnicas utilizadas na visão computacional, e algumas delas merecem destaque. Uma delas é a utilização de redes neurais convolucionais (CNNs), que se tornaram um dos principais pilares dessa área. Outra técnica importante é a estereovisão, que permite a obtenção de informações de profundidade a partir de duas ou mais imagens capturadas por câmeras diferentes.

Além disso, a visão computacional também se beneficia de técnicas de processamento de imagens, como filtragem, transformações geométricas, correção de distorção, entre outras. Essas técnicas permitem melhorar a qualidade das imagens, remover ruídos e realizar pré-processamento antes da análise computacional.

2.2.1 Imagens Digitais

Imagens digitais são representações visuais armazenadas e processadas em formato digital. Elas são compostas por uma grade de elementos individuais chamados de pixels, que são os blocos fundamentais que compõem a imagem. Cada pixel possui uma localização específica na grade e um valor que determina sua cor e intensidade (18).

A digitalização de imagens analógicas, como fotografias ou desenhos, é um processo pelo qual a informação visual é convertida em uma representação digital. Isso permite que as imagens sejam armazenadas, transmitidas, modificadas e exibidas eletronicamente em dispositivos como computadores, smartphones e tablets.

A imagem digital é uma peça central na aplicação prática da visão computacional. Por meio

da representação numérica das informações visuais, as imagens digitais possibilitam a análise e interpretação computacional do conteúdo visual. Essa interseção entre imagem digital e visão computacional é fundamental para uma variedade de tarefas, como detecção e reconhecimento de objetos, segmentação de imagens, rastreamento de movimento e reconhecimento facial.

A imagem digital, portanto, desempenha um papel crucial na capacidade dos computadores de compreender e tomar decisões com base em informações visuais, impulsionando avanços em diversas áreas de aplicação.

2.2.2 Reconhecimento de expressões faciais

Os seres humanos são capazes de reconhecer milhares de rostos ao longo dos anos, mesmo quando o estímulo visual muda, como quando as pessoas fazem expressões faciais, envelhecem ou usam óculos, barba ou fazem mudanças no penteado. No entanto, é uma tarefa difícil desenvolver um modelo computacional capaz de reconhecer rostos. Isso ocorre porque os rostos são estímulos visuais complexos e multidimensionais. O reconhecimento de rostos é uma tarefa de visão computacional de alto nível (1).

Aqui estão alguns dos desafios do reconhecimento de rostos:

- Os rostos podem variar muito em aparência, dependendo da etnia, idade, sexo e outros fatores;
- Os rostos podem ser vistos em diferentes ângulos e iluminação;
- Os rostos podem ser parcialmente ocultos por objetos, como óculos ou barba;
- Os rostos podem ser alterados artificialmente, como com Photoshop.

Apesar desses desafios, o reconhecimento de rostos é uma área de pesquisa ativa com muitas aplicações potenciais. Por exemplo, o reconhecimento de rostos pode ser usado para Segurança, Marketing e Medicina.

Stan e Anil (21) argumentam que o aumento no número de computadores com maior capacidade de processamento e baixo custo levou a um aumento no interesse no processamento digital de imagens. Isso, por sua vez, levou ao desenvolvimento de novas aplicações de processamento de imagens, como autenticação biométrica e vigilância. Todas essas aplicações têm o reconhecimento de face como uma parte primordial.

Agarwal (1) descreve dois métodos básicos para reconhecimento de rosto. O primeiro método é baseado na extração de vetores de características das partes básicas de um rosto, como olhos, nariz, boca e queixo. Esses vetores são então convertidos em um único vetor de recursos. O segundo método é baseado na análise de componentes principais (PCA). A PCA é uma técnica estatística que identifica os componentes principais de um conjunto de dados. No caso do reconhecimento de rosto, os componentes principais são as características que melhor descrevem uma face.

Ambos os métodos têm suas próprias vantagens e desvantagens. O método baseado em modelos deformáveis é mais robusto a variações na pose e iluminação. No entanto, ele é mais complexo e requer mais poder de processamento. O método baseado na PCA é menos complexo e requer menos poder de processamento. No entanto, ele é menos robusto a variações na pose e iluminação.

2.3 Aprendizado de máquina (Machine Learning)

Murphy (26) define Aprendizado de Máquina como um conjunto de métodos que podem detectar padrões em dados automaticamente e, em seguida, usar os padrões descobertos para prever dados futuros ou para realizar outros tipos de tomada de decisão sob incerteza, como planejar como coletar mais dados.

2.3.1 Categoria de Modelos de Aprendizado de Máquina

Pode ser dividido em três categorias, levando em conta quais problemas quer resolver:

- **Aprendizado Supervisionado:** Este tipo de aprendizado é o mais comum. Os modelos de aprendizado supervisionado são treinados em dados que já estão rotulados, o que significa que cada exemplo de dados tem uma etiqueta correspondente que diz ao modelo o que ele deve fazer. Por exemplo, um modelo de aprendizado supervisionado pode ser treinado em um conjunto de dados de imagens de cães e gatos, com cada imagem rotulada como um cão ou um gato. Depois de treinado, o modelo será capaz de identificar cães e gatos em novas imagens.
- **Aprendizado não Supervisionado:** Este tipo de aprendizado não requer dados rotulados. Os modelos de aprendizado não supervisionado são capazes de encontrar padrões nos dados sem saber o que significam. Por exemplo, um modelo de aprendizado não supervisionado pode ser usado para agrupar clientes com base em seus hábitos de compra.
- **Aprendizado por Reforço:** Este tipo de aprendizado envolve o uso de recompensas e punições para ensinar uma máquina a realizar uma tarefa. Por exemplo, um robô pode ser treinado a jogar um jogo de tênis usando o aprendizado por reforço. O robô seria recompensado por acertar a bola e punido por errar a bola. Com o tempo, o robô aprenderia a jogar o jogo de tênis de forma mais eficaz.

2.3.2 Modelos de Aprendizado de Máquina

2.3.2.1 Classificação

A classificação é um método de predição de uma classe discreta ou categoria. Por exemplo, existe um conjunto de dados de carros, com cada carro rotulado como sedan, hatch ou SUV. É

possível usar um algoritmo de classificação para aprender as características que definem cada tipo de carro. Em seguida, pode-se usar esse conhecimento para classificar novos carros.

Alguns algoritmos de classificação comuns incluem:

- **K-nearest neighbors (KNN):** esse algoritmo classifica um novo carro com base nos carros mais próximos dele no conjunto de dados.
- **Árvores de Decisão:** esse algoritmo classifica um novo carro usando uma árvore de decisões, onde cada nó da árvore representa uma característica do carro.
- **Support vector machines (SVMs):** esse algoritmo classifica um novo carro usando um hiperplano, que é uma linha imaginária que separa os carros de diferentes categorias.
- **Regressão Logística:** esse algoritmo classifica um novo carro atribuindo-lhe uma probabilidade de pertencer a cada categoria.

2.3.2.2 Regressão

A regressão é um método de previsão de um valor contínuo. Por exemplo, uma pesquisa para saber o valor de um imóvel e melhor momento para investir. É possível usar a regressão para prever o valor do imóvel no futuro. Pode-se também pode usar a regressão para prever qual bairro terá o maior crescimento urbano.

Algumas técnicas de regressão são:

- **Regressão linear simples:** esse tipo de regressão usa apenas uma variável independente para prever a variável dependente. Por exemplo, você pode usar a regressão linear simples para prever o valor de uma casa com base na sua área.
- **Regressão linear múltipla:** esse tipo de regressão usa várias variáveis independentes para prever a variável dependente. Por exemplo, você pode usar a regressão linear múltipla para prever o valor de uma casa com base na sua área, no número de quartos e no número de banheiros.
- **Regressão não linear:** esse tipo de regressão usa uma função não linear para prever a variável dependente. Por exemplo, você pode usar a regressão não linear para prever o preço de uma ação com base em suas taxas de crescimento.

2.3.2.3 Agrupamento

O agrupamento é um método de classificação de dados em grupos semelhantes. O objetivo do agrupamento é encontrar grupos de dados que sejam semelhantes entre si e diferentes de outros grupos de dados.Existem muitos algoritmos de agrupamento diferentes, cada um com suas próprias vantagens e desvantagens. Alguns dos algoritmos de agrupamento mais comuns: K-means, Mean-Shift, DBSCAN, Single Linkage e Complete Linkage.

O algoritmo de agrupamento mais adequado para uma tarefa específica dependerá de uma variedade de fatores, incluindo o tipo de dados, o número de grupos e a quantidade de precisão desejada. Alguns exemplos de como o agrupamento pode ser usado são: Segmentação de mercado, Compactação de imagem, Análise e rotulagem de novos dados, entre outros.

2.3.2.4 Associação

As regras de associação são um tipo de aprendizado de máquina que é usado para encontrar padrões em dados. As regras de associação são frequentemente usadas para analisar carrinhos de compras, automatizar estratégias de marketing e outras tarefas relacionadas a eventos.

2.4 Aprendizado Profundo (Deep Learning)

Deep learning, ou aprendizado profundo, é um subcampo da aprendizagem de máquina que se concentra em treinar redes neurais artificiais com múltiplas camadas para aprender representações hierárquicas de dados (23). Isso revolucionou diversas áreas, incluindo visão computacional, processamento de linguagem natural e reconhecimento de fala.

O bloco fundamental do deep learning é a Rede Neural Artificial, inspirada na estrutura e função do cérebro humano. Essas redes consistem em nós interconectados, chamados neurônios ou unidades, organizados em camadas. Cada neurônio recebe entradas da camada anterior, aplica uma transformação não linear à elas e repassa o resultado para a próxima camada. Redes neurais profundas geralmente possuem muitas camadas ocultas, permitindo que elas aprendam representações cada vez mais abstratas e complexas dos dados de entrada (19).

O aprendizado profundo é uma técnica de aprendizado de máquina que pode aprender automaticamente recursos ou representações a partir de dados brutos. Isso elimina a necessidade de recursos projetados manualmente, que podem ser difíceis e demorados de criar. A capacidade de aprendizado de recursos do aprendizado profundo levou a avanços nas tarefas de visão computacional, como classificação de imagens, detecção de objetos e segmentação de imagens. Modelos de aprendizado profundo, como redes neurais convolucionais (CNNs), alcançaram uma precisão sem precedentes nessas tarefas, aprendendo recursos hierárquicos a partir de grandes quantidades de dados rotulados.

Essa técnica depende muito de conjuntos de dados em larga escala e recursos computacionais poderosos, especialmente unidades de processamento gráfico (GPUs), para treinar modelos complexos de maneira eficiente (19). O treinamento de redes neurais profundas envolve alimentar o modelo com dados rotulados e ajustar iterativamente os pesos da rede por meio de um processo chamado retropropagação. A retropropagação calcula o gradiente da função de perda em relação aos parâmetros da rede, permitindo que o modelo atualize seus pesos e melhore seu desempenho ao longo do tempo.

2.4.1 Modelos de Aprendizado Profundo

Existem vários modelos de deep learning que foram desenvolvidos para diferentes tarefas e aplicações. Alguns dos modelos mais populares e influentes:

- **Redes Neurais Convolucionais (Convolutional Neural Networks - CNNs):** São amplamente usadas em tarefas de visão computacional, como classificação de imagens, detecção de objetos e segmentação de imagens. As CNNs serão objeto de estudo deste trabalho logo serão mais detalhadas mais pra frente.
- **Redes Neurais Recorrentes (Recurrent Neural Networks - RNNs):** São utilizadas em tarefas de processamento de sequências, como tradução automática, geração de texto e reconhecimento de fala. As RNNs possuem conexões retroalimentadas, permitindo que informações contextuais sejam mantidas em memória para processar sequências de dados.
- **Redes Generativas Adversariais (Generative Adversarial Networks - GANs):** São compostas por duas redes neurais: o gerador e o discriminador. O gerador gera amostras sintéticas que tentam enganar o discriminador, enquanto o discriminador tenta distinguir entre as amostras reais e as sintéticas. Essa competição entre as redes leva ao aprendizado de um modelo gerador capaz de criar amostras realistas.

2.5 Redes Neurais de Convolução (CNNs)

As Redes Neurais Convolucionais são um tipo especializado de modelo de deep learning amplamente utilizado em tarefas de visão computacional. Elas foram desenvolvidas para lidar com dados de imagem, aproveitando a estrutura espacial e hierárquica das informações visuais.

Uma característica fundamental das CNNs é a camada convolucional, que realiza operações de convolução nas imagens de entrada. A convolução envolve a aplicação de filtros ou kernels espaciais em pequenas regiões da imagem, permitindo que a rede extraia características localizadas, como bordas, texturas e padrões específicos (3). Essa operação é repetida em toda a imagem, gerando mapas de características convolucionais.

A grande vantagem das CNNs é sua capacidade de aprender representações hierárquicas e invariantes de características visuais. À medida que as informações passam pelas camadas convolucionais, a rede aprende a reconhecer padrões cada vez mais complexos e abstratos, capturando relações espaciais e semânticas nas imagens. Essa capacidade de aprendizado de recursos hierárquicos é fundamental para o sucesso das CNNs em tarefas como classificação de imagens, detecção de objetos e segmentação semântica.

Uma Rede Neural Convolucional (CNN) típica é composta por três tipos principais de camadas: a camada convolucional (convolutional layer), a camada de pooling máxima (maxpooling layer) e a camada totalmente conectada (fully connected layer) (3). Essas camadas desempenham papéis distintos na extração e processamento das características das imagens durante o treinamento da CNN.

2.5.1 Camada de convolução (Convolutional layer)

A camada convolucional é responsável por realizar operações de convolução nas imagens de entrada. Ela consiste em um conjunto de filtros ou kernels espaciais que são aplicados a pequenas regiões das imagens. Cada filtro detecta características específicas, como bordas, texturas ou padrões relevantes na imagem. Durante a convolução, o filtro desliza por toda a imagem, multiplicando os valores dos pixels pelos pesos do filtro em cada posição e somando-os para gerar um mapa de características, como mostra a Figura 2.1. Essa operação é repetida para cada filtro, resultando em vários mapas de características. A camada convolucional ajuda a extrair características localizadas e espaciais das imagens, aprendendo representações cada vez mais complexas à medida que as informações são processadas em camadas subsequentes.

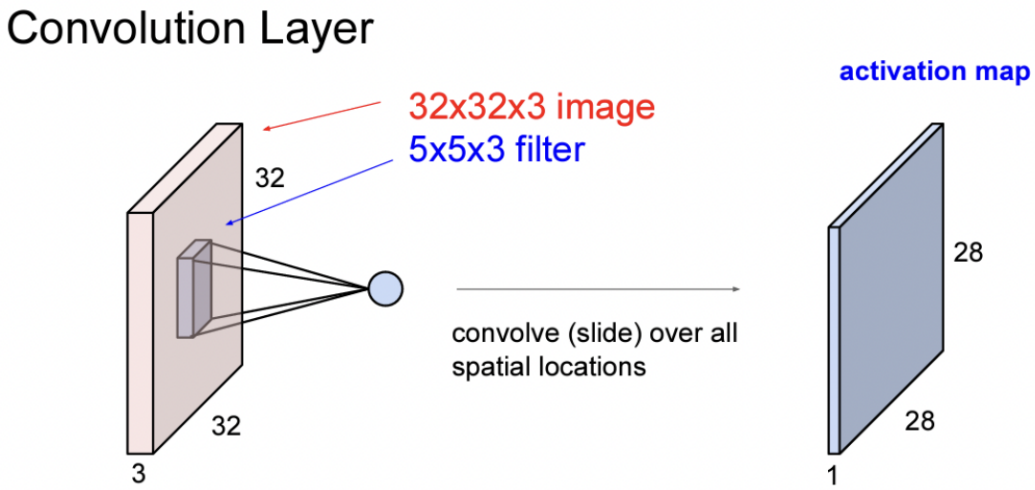

Figura 2.1: Exemplificacão da operação de convolução
Fonte: Lecture 5: Convolutional Neural Networks (24)

2.5.2 Camada de pooling (Maxpooling layer)

A camada de pooling máxima é usada para reduzir a dimensionalidade dos mapas de características gerados pelas camadas convolucionais. Ela opera dividindo a imagem em regiões não sobrepostas e selecionando o valor máximo em cada região, como mostra a Figura 2.2. Essa operação tem como objetivo capturar as características mais dominantes da região e reduzir a quantidade de dados a serem processados nas camadas subsequentes. O pooling máxima ajuda a tornar a CNN mais robusta a pequenas variações na posição ou escala das características aprendidas, permitindo uma representação mais compacta e invariante das características relevantes.

2.5.3 Camada não linear (Fully connected layer)

A camada totalmente conectada é responsável pela classificação final das características extraídas da imagem. Ela é semelhante às camadas ocultas de uma rede neural tradicional. Nessa

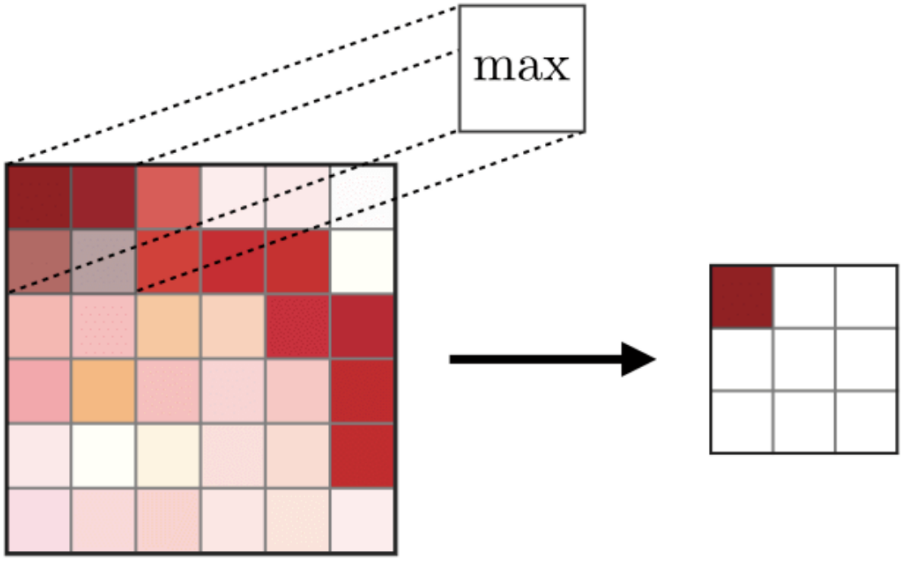

Figura 2.2: Exemplificacão da operação de max-pooling
Fonte: Convolutional Neural Networks Cheatsheet (2)

camada, cada unidade (ou neurônio) está conectada a todas as unidades da camada anterior. Os valores de saída dessas unidades são ponderados por pesos e passam por funções de ativação para gerar as previsões finais da CNN. Essas previsões podem ser as probabilidades de classificação em tarefas de classificação de imagens, por exemplo. A camada totalmente conectada integra as características extraídas pelas camadas convolucionais e de pooling para produzir uma saída final que representa a classificação ou a regressão desejada.

2.6 Fine-tuning

O fine-tuning, também conhecido como aprendizado por transferência, desempenha um papel crucial na obtenção de um bom desempenho da rede proposta. Essa técnica envolve a adaptação do conhecimento prévio desenvolvido por um modelo para tarefas específicas, economizando tempo e recursos computacionais (33).

O processo de fine-tuning começa com um modelo base como ponto de partida e envolve o ajuste refinado de diversos metaparâmetros que impactam o desempenho da rede proposta. O objetivo é adaptar as características do modelo da melhor forma possível para a tarefa específica da aplicação, aproveitando o conhecimento prévio e, consequentemente, melhorando o desempenho.

Os metaparâmetros são as propriedades da rede que devem ser testadas, avaliadas e selecionadas com o objetivo de obter os melhores resultados (6). Durante esse processo, serão realizados testes e avaliações dos diversos metaparâmetros da rede, buscando encontrar as melhores atribuições possíveis.

Abaixo alguns metaparâmetros utilizados nesta dissertação:

2.6.1 Taxa de Aprendizado (Learning Rate)

A taxa de aprendizado controla a magnitude dos ajustes feitos nos pesos da rede durante o processo de otimização. Um valor muito baixo pode resultar em um treinamento lento, enquanto um valor muito alto pode fazer com que o treinamento oscile ou não convirja. A escolha adequada da taxa de aprendizado é essencial para um treinamento eficaz da CNN.

2.6.2 Épocas (Epochs)

Uma época é definida como uma iteração completa sobre o conjunto de treinamento. O número de épocas especifica quantas vezes a CNN passará por todo o conjunto de treinamento. Treinar por muitas épocas pode levar ao superajustamento (overfitting), enquanto treinar por poucas épocas pode resultar em um modelo subajustado (underfitting).

2.6.3 Função de Ativação

A função de ativação é aplicada após cada camada convolucional ou totalmente conectada da CNN. Ela introduz não-linearidades na rede, permitindo que a CNN aprenda representações complexas dos dados. A escolha da função de ativação pode afetar a capacidade da CNN de aprender e representar padrões nos dados.

- **ReLU:** ReLU é uma função de ativação amplamente utilizada em redes neurais, incluindo Convolutional Neural Networks (CNNs). A função ReLU é definida como f(x) = max(0, x), onde x é a entrada para a função. Em outras palavras, a função ReLU retorna zero para todos os valores negativos de x e retorna x para valores não negativos.

  Uma das principais vantagens da função ReLU é sua capacidade de lidar com o problema do desaparecimento do gradiente. Durante o treinamento de uma rede neural profunda, o gradiente pode diminuir significativamente à medida que se propaga para camadas anteriores, tornando o aprendizado mais difícil. No entanto, a função ReLU mantém o gradiente constante para valores positivos de x, permitindo que o aprendizado ocorra de forma mais eficaz.

- **Softmax:** Softmax é uma função de ativação utilizada principalmente na camada de saída de uma CNN para problemas de classificação multiclasse. A função Softmax transforma um vetor de valores reais em um vetor de probabilidades, onde cada valor de saída representa a probabilidade de pertencer a uma determinada classe.

  A função Softmax é definida como:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.1}$$

Onde Zi é o valor de entrada para a função Softmax e K é o número total de classes.

2.6.4 Optimizer function

Referem-se aos algoritmos e técnicas utilizadas para ajustar os pesos das conexões entre os neurônios em uma rede neural durante o treinamento. O objetivo é minimizar a função de perda, que mede a diferença entre as saídas da rede e os rótulos corretos dos dados de treinamento.

Nesta dissertação foram usadas as seguintes técnicas:

2.6.4.1 ADAM (Adaptive Moment Estimation)

Foi proposto por Diederik P. Kingma e Jimmy Ba em 2015. O Adam é considerado um algoritmo adaptativo, pois ajusta a taxa de aprendizado para cada parâmetro individual com base em estimativas de momento de primeira e segunda ordem dos gradientes. O Adam combina os benefícios do método RMSprop, que ajuda a controlar a taxa de aprendizado para cada parâmetro individualmente, e do Momentum, que acelera o processo de aprendizado, permitindo que ele se mova mais rapidamente na direção do mínimo. Além disso, o Adam realiza a correção do viés inicial nos momentos m e v, o que contribui para uma convergência mais rápida e estável.

Ele usa estimativas dos momentos do gradiente de primeira e segunda ordem para adaptar a taxa de aprendizado para cada parâmetro. Essa abordagem adaptativa permite que o algoritmo otimize eficientemente uma ampla variedade de problemas de aprendizado de máquina, tornando-o uma escolha comum em muitas implementações de redes neurais.

2.6.4.2 SGD (Stochastic Gradient Descent)

É uma versão estocástica do algoritmo clássico de otimização Gradient Descent (Descida do Gradiente). A principal diferença entre o SGD e o Gradient Descent tradicional é que o SGD atualiza os parâmetros da rede após cada exemplo de treinamento, em vez de após todo o conjunto de treinamento.

Este algoritmo atualiza os parâmetros do modelo após cada pequeno lote de exemplos de treinamento, o que o torna eficiente em termos de memória e computação. No entanto, a oscilação e a variabilidade podem ser desafios a serem enfrentados, especialmente em problemas de otimização complexos.

2.6.4.3 RMSprop (Root Mean Square Propagation)

Uma extensão do algoritmo de otimização Gradient Descent (Descida do Gradiente) que aborda alguns dos desafios associados à convergência em problemas de otimização não convexas. O algoritmo utiliza uma média móvel exponencial dos quadrados dos gradientes anteriores para adaptar a taxa de aprendizado para cada parâmetro individualmente. Essa abordagem adaptativa ajuda

a melhorar a eficiência e a convergência do processo de otimização, tornando o RMSprop uma escolha comum em muitas implementações de aprendizado de máquina.

2.6.5 Neurônios

Os neurônios, também conhecidos como unidades ou células, desempenham um papel fundamental em modelos de Aprendizado de Máquina Profundo. São unidades computacionais que recebem um conjunto de entradas, realizam cálculos ponderados e aplicam uma função de ativação para produzir uma saída. A inspiração para a estrutura dos neurônios provém do funcionamento dos neurônios biológicos do cérebro humano (19).

Em uma rede neural, cada neurônio em uma camada está conectado a todos os neurônios da camada anterior. Essas conexões são denominadas pesos da rede e são ajustadas durante o processo de treinamento para que o modelo aprenda a mapear corretamente os padrões nos dados de entrada para as saídas desejadas.

A variação do número de neurônios em uma camada da rede tem impacto direto na capacidade de representação, complexidade computacional, regularização e eficiência de treinamento do modelo. Portanto, a escolha do número ideal de neurônios deve ser ajustada especificamente para cada modelo de Aprendizado de Máquina Profundo.

2.7 Sistema de Múltiplos Classificadores

Sistemas de Múltiplos Classificadores (MCS, do inglês Multiple Classifier Systems) são abordagens que envolvem a combinação de múltiplos classificadores para realizar tarefas de classificação. A ideia fundamental dos MCS é que a combinação dos resultados de classificadores individuais pode levar a um desempenho geral superior em comparação com o uso de um único classificador. Os MCS precisam conter diversidade por meio de métodos como: diferentes conjuntos de dados para treinar os modelos; um mesmo modelo de classificação, porém com diferentes configurações/-parâmetro ou; diferentes tipos de classificadores (4).

A motivação por trás dos MCS está enraizada no princípio da diversidade. Ao utilizar classificadores diferentes, que são treinados com diferentes algoritmos, parâmetros ou conjuntos de dados, é possível obter uma variedade de perspectivas sobre o problema de classificação. Cada classificador pode ter pontos fortes e fracos em diferentes partes do espaço de características, tornando a combinação de suas decisões uma estratégia promissora. A combinação de classificadores em Sistemas de Múltiplos Classificadores (MCS) é motivada por três fatores principais: a motivação estatística, a motivação representacional e a motivação computacional (16) (11). Essas motivações fornecem uma base sólida para a utilização de abordagens de combinação de classificadores.

A **motivação estatística** destaca que a combinação de diferentes classificadores pode reduzir a variância dos resultados e melhorar a estabilidade das predições. Ao combinar as decisões de vários classificadores independentes, é possível reduzir a incerteza e obter uma estimativa mais precisa da classe verdadeira. Essa abordagem estatística permite uma melhor confiabilidade e robustez do

sistema de classificação.

A **motivação representacional** enfatiza a diversidade e complementaridade dos classificadores. Ao utilizar diferentes algoritmos de aprendizado ou conjuntos de características, é possível obter diferentes perspectivas sobre os dados. Cada classificador pode se especializar em detectar certas características ou padrões, enquanto pode ser menos eficaz em outros. Ao combinar os resultados desses classificadores, é possível aproveitar a variedade de conhecimentos representacionais e melhorar a capacidade geral de discriminação do sistema (16).

A **motivação computacional** aborda a questão do custo computacional e do tempo de execução. Em alguns casos, a combinação de classificadores pode ser computacionalmente mais eficiente do que o treinamento e a execução de um único classificador complexo (16). Ao distribuir a tarefa de classificação entre múltiplos classificadores, é possível reduzir a carga computacional individual e acelerar o processo de classificação.

A construção de um Sistema de Múltiplos Classificadores (MCS) pode ser dividida em duas etapas principais: a construção do comitê e o processo de combinação (4). A etapa de construção do comitê envolve a criação de um conjunto de modelos, também conhecido como pool de classificadores, garantindo a diversidade dentro desse conjunto. Na etapa de processo de combinação, os resultados dos diferentes modelos são integrados para gerar uma classificação final (32). Existem vários métodos para realizar essa combinação, como a média dos resultados ou o voto majoritário.

Essas abordagens de construção do comitê e de combinação dos resultados permitem que um MCS aproveite a diversidade dos classificadores para melhorar o desempenho e a robustez da classificação final.

2.8 Análises Estatísticas

A escolha do teste estatístico para analisar a diferença entre amostras depende da verificação da normalidade dos dados. Se os dados seguirem uma distribuição normal, utiliza-se um teste paramétrico. Caso contrário, recorre-se a um teste não paramétrico como alternativa.

2.8.1 Shapiro-Wilk

O Teste de Shapiro é uma ferramenta estatística utilizada para verificar se uma amostra de dados segue uma distribuição normal ou não. A utilização do Teste de Shapiro envolve três partes principais:

1. Elaboração das hipóteses
    - H_0: a amostra provém de uma população normal
    - H_1: a amostra não provém de uma população normal
2. Definição do nível de confiança do teste

- $\alpha = 0.05$

3. Tomada de Decisão referente ao *p-value*

    - Não é possível rejeitar H_0 se *p-value* $> \alpha$
    - Rejeita H_0 se *p-value* $< \alpha$

Se a amostra rejeitar a hipótese nula (H_0), será necessário utilizar um teste não paramétrico para a comparação das amostras. Neste estudo, o *teste de Wilcoxon* será empregado. Por outro lado, se não houver evidência suficiente para rejeitar a hipótese nula (H_0), será adequado empregar um teste paramétrico. Neste caso, o *teste t-Student* será utilizado no estudo.

Capítulo 3

Trabalhos Relacionados

Este capítulo trata os trabalhos relacionados a construção de um Sistema de Múltiplo Classificadores para o aumento da precisão na classificação de emoções.

3.1 Exploiting Convolutional Neural Networks to Recognize Emotions

O estudo realizado por Filho et al (14) tem como objetivo abordar duas questões principais: a possibilidade de classificar emoções em imagens usando uma rede neural convolucional (CNN) e qual seria a arquitetura mais adequada considerando diferentes metaparâmetros. O experimento envolveu 10 iterações, onde o "Learning Rate"e o "Update Algorithms"(ou "Optimizer Function") foram variados, buscando encontrar a combinação de metaparâmetros que produzisse os melhores resultados. Em seguida, com base nessa melhor combinação, foram realizadas mais 5 iterações, variando os neurônios que compunham a camada de "Multilayer Perceptron"(MLP). Além disso, dois outros modelos, AlexNet e LeNet5, foram utilizados para validar os resultados obtidos.

No entanto, este estudo não implementou a proposta desta dissertação, que consiste no uso de um sistema de classificadores, o que ocasionaria em um aumento da precisão do modelo.

3.2 A Low-Cost Smart Home Automation to Enhance Decision-Making based on Fog Computing and Computational Intelligence

O trabalho de (22) apresenta uma solução para automação residencial chamada STORm. STORm é um comitê de classificadores que utiliza o paradigma de Fog Computing. O trabalho usou algoritmos renomados e clássicos, como Naive Bayes, SVM e KNN, para criar o comitê de classificadores. Os dados utilizados para os treinamentos foram coletados de aparelhos conectados à residência por um período de 40 dias. O trabalho comparou o resultado de cada classificador e

também do comitê inteiro para comprovar o aumento da precisão. Os resultados mostraram que o comitê de classificadores teve uma precisão maior do que qualquer classificador individual.

Um trabalho muito semelhante no que se trata de arquitetura, porém em uma área diferente, visto que essa dissertação usa como dados as imagens digitais representando as emoções. Imagens digitais são informações mais volumosas do que os dados utilizados no trabalho de e isso traz desafios diferentes para a implementação do sistema comitê.

3.3 Avaliação de Comitês com Classificadores Tradicionais e Profundos para Análise de Sentimentos

O projeto (4) tem como objetivo avaliar a acurácia de alguns classificadores individuais em comparação ao comitê de classificadores para análise de sentimentos em textos. Além disso, foram usadas dois tipos de base, uma binária (positiva/negativa) e uma multiclasses. É importante salientar que este projeto tem semelhanças com a dissertação sobre imagens digitais, mas difere no fato de criar um comitê que avalia sentimento em textos e não em imagens digitais.

3.4 Sistema de Reconhecimento de Expressões Faciais para Classificação de Emoções de Usuários em Sistemas Computacionais

O trabalho de Mendonça (25) propõe um sistema automatizado capaz de receber os dados gerados durante a interação do usuário com o sistema computacional, como imagens e vídeos capturados nesse processo, e aplicar técnicas de reconhecimento facial e de expressões faciais. As expressões são categorizadas em alegria, desgosto, raiva e surpresa, e para a classificação, é utilizada uma Rede Neural Convolucional treinada com o conjunto de dados aumentado da base JAFFE.

Os resultados obtidos são promissores, alcançando uma taxa de acerto de 62%, considerando a complexidade da tarefa e o baixo número de imagens disponíveis para treinamento. Esse avanço contribui significativamente para a compreensão das interações humano-computador e pode abrir caminho para melhorias no design de interfaces e na experiência do usuário.

3.5 RedFace – Um Sistema de Reconhecimento de Expressões Faciais para Apoiar um Ambiente Virtual de Aprendizagem

Diniz (12) propõe melhorias no processo de aprendizagem e nas atividades avaliativas de um AVA que utiliza agentes inteligentes. O sistema proposto incorpora técnicas avançadas de reconhecimento facial, incluindo Análise dos Componentes Principais (PCA) para identificação facial e um Sistema de Codificação de Ação Facial (FACS) para rastreamento de pontos característicos faciais. Além disso, é empregado um método de classificação de emoções com base em regras para o reconhecimento de expressões faciais.

Com o auxílio de uma câmera, o sistema pode identificar o aluno durante todas as avaliações pedagógicas do AVA, além de reconhecer a emoção (alegria, tristeza, raiva e desgosto) do aluno com base em sua expressão facial. Essas informações sobre a emoção do aluno são utilizadas como entrada para um agente pedagógico animado presente no AVA. O agente, por sua vez, busca motivar o aluno de forma mais eficaz, promovendo uma experiência de aprendizagem mais personalizada e engajadora.

3.6 A sentiment classification model based on multiple classifiers

O objetivo do estudo (28) é investigar o potencial benefício do conceito de sistemas de classificação múltipla no problema de classificação de sentimento em turco e propor uma nova técnica de classificação. O algoritmo de Voto Majoritário foi usado em conjunto com três classificadores: Naive Bayes, Máquina de Vetores de Suporte (SVM) e Bagging. Os parâmetros do SVM foram otimizados quando usado como um classificador individual. Os resultados experimentais mostraram que os sistemas de classificação múltipla aumentam o desempenho dos classificadores individuais nos conjuntos de dados de classificação de sentimento em turco e os meta classificadores contribuem para a eficácia desses sistemas de classificação múltipla.

A abordagem proposta obteve melhor desempenho do que o Naive Bayes, que havia sido relatado como o melhor classificador individual para esses conjuntos de dados, e as Máquinas de Vetores de Suporte. Os sistemas de classificação múltipla (MCS) são uma boa abordagem para a classificação de sentimento, e a otimização dos parâmetros dos classificadores individuais deve ser levada em conta ao desenvolver sistemas de previsão baseados em MCS.

Capítulo 4

Metodologia proposta

Este capítulo apresenta de forma teórica a problemática da pesquisa, uma sistematização acerca do problema, detalha de forma técnica uma visão geral da solução proposta, arquitetura da solução proposta e aprofunda-se na criação dos Classificadores Individuais.

4.1 Análise do problema

O processo de reconhecimento e classificação de emoções humanas a partir de imagens digitais é um tópico muito explorado na literatura, mas segue sendo um desafio complexo na área de IA. Tradicionalmente, algoritmos de IA utilizam técnicas de processamento de imagens e extração de características para tentar identificar as emoções expressas nas imagens. No entanto, esses algoritmos tradicionais enfrentam dificuldades em relação ao custo computacional pois exigem etapas de pré-processamento intensivas. Essas etapas consomem tempo e recursos computacionais significativos, o que torna o processo lento e pouco viável para aplicação em tempo real ou em grande escala.

Com o rápido advento da área de AM novos algoritmos de Aprendizado Profundo surgiram, como as CNNs que possuem uma capacidade inata de aprender características diretamente dos dados brutos, muitas vezes sem a necessidade de um pré-processamento manual. Tais algoritmos de Aprendizado Profundo são treinados em grandes conjuntos de dados, em que cada imagem é associada a uma emoção específica.

Trabalhos na literatura também apontam as técnicas de MCS que utilizam a combinação de múltiplos classificadores individuais para melhora na precisão e generalização de um modelo de aprendizado de máquina. Em vez de depender de um único classificador para tomada de decisões, o MCS utilizam um conjunto de classificadores independentes e combinam suas predições para obter um resultado final.

Visando resolver a problemática da ineficácia do uso dos algoritmos tradicionais de AM, a solução proposta de determinar Classificadores Individuais baseados em modelos de CNNs por um processo de *"finetuning"* obtendo a melhor arquitetura e metaparâmetros da rede para o caso

específico de reconhecimento e classificação de emoções humanas a partir de imagens digitais. E seguinte desenvolvimento de um Sistema Comitê denominado de *"SISCOM"*, utilizando-se dos múltiplos Classificadores Individuais obtidos anteriormente, combinando as predições dos vários classificadores independentes com o objetivo de melhorar a precisão e generalização da rede.

4.2 Visão geral do SISCOM

A solução proposta SISCOM é baseada no desenvolvimento e implementação de um Sistema Comitê ou MCS, visando uma melhoria na precisão e generalização da rede proposta para o processo de reconhecimento e classificação de emoções humanas básicas a partir de imagens digitais.

O desenvolvimento da solução SISCOM, seguiu as duas etapas principais da construção de um MCS: (etapa de construção do comitê) e (etapa processo de combinação) (4). A primeira etapa de construção do comitê consiste em criar o conjunto dos múltiplos Classificadores Individuais denominado como "Pool de Classificadores".

Já a segunda etapa, consiste no processo de combinação das predições geradas pelos classificadores independentes para a tomada de decisão em uma classificação final. A integração dos diferentes resultados obtidos pelos modelos de Classificadores Individuais pode ser obtida por variados métodos, aplicando métodos estatísticos de média ponderada ou pelo voto majoritário.

Para a solução SISCOM, optou-se pela escolha da média ponderada de forma que as predições de cada classificador são ponderadas de acordo com sua confiabilidade determinada pela performance obtida individualmente, em seguida, as predições são combinadas por meio de uma média ponderada em uma classificação final. A Equação 4.1, a seguir demonstra o processo de combinação pela média ponderada.

$$x_f = \frac{1}{n} \sum_{i=1}^{n} w(i).x_i \tag{4.1}$$

Um dos objetivos estabelecidos dessa dissertação foi de determinar o melhor número de Classificadores Individuais integrados para composição da solução proposta SISCOM. Dessa forma, o método científico que será aplicado para atingir esse objetivo é o desenvolvimento de dois Sistemas Comitês *(SISCOM I e SISCOM II)*.

Visando testar e analisar duas configurações diferentes, para o caso da solução SISCOM I serão integrados um total de três modelos de Classificadores Individuais e para a solução SISCOM II serão integrados um total de cinco modelos de Classificadores Individuais. Dessa forma, a expectativa é poder comparar a performance e resultados das duas configurações e determinar a influência da integração de números diferentes de modelos de Classificadores Individuais.

4.3 Arquitetura do SISCOM

A arquitetura proposta para a solução SISCOM, se divide em três módulos bem definidos:

- O primeiro módulo denominado *"Módulo dataset"*, refere-se à consolidação das bases de dados que serão utilizadas para compor o dataset final;
- O segundo módulo denominado *"Módulo pool de classificadores"* é o conjunto dos múltiplos Classificadores Individuais desenvolvidos que irão utilizar o dataset final para treinamento dos próprios modelos;
- O terceiro e último módulo denominado *"Módulo de decisão"* é responsável pela integração e combinação das predições dos múltiplos Classificadores Individuais em uma classificação final utilizando o método estatístico de média ponderada.

A opção pela divisão em módulos, deu-se também visando facilitar a implementação da solução proposta. Assim, segmentando a implementação também em três etapas bem definidas possibilitando focar os esforços no desenvolvimento de cada módulo e por fim realizar a integração dos módulos gerando a solução final SISCOM. A Figura 4.1 a seguir, detalha a arquitetura geral do SISCOM.

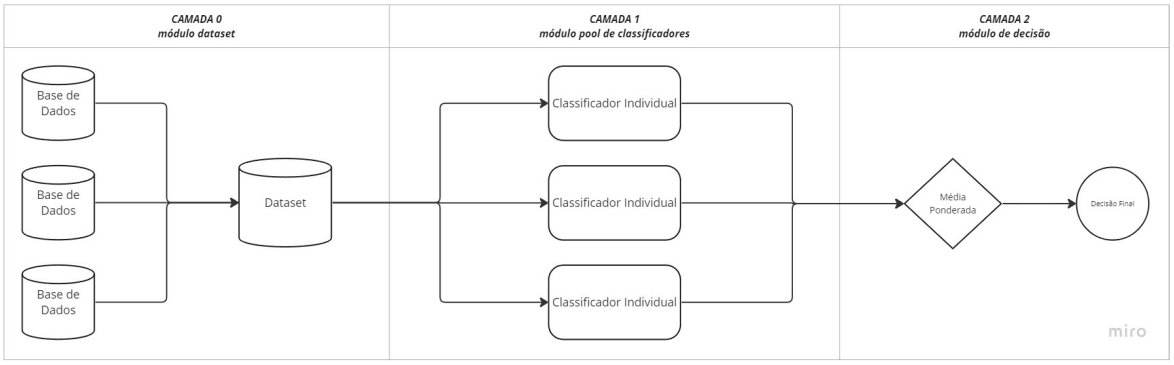

Figura 4.1: Arquitetura da solução SISCOM

Fonte: Autor

Um ponto a se destacar do (Módulo pool de classificadores) da arquitetura detalhada da solução SISCOM é a implementação em paralelo dos múltiplos Classificadores Individuais. Uma decisão visando otimizar o custo computacional do SISCOM, por tanto, o treinamento dos modelos dos Classificadores Individuais é feito de forma independente, evitando um gargalo e por fim apenas utilizando-se das predições de cada modelo para realizar a combinação e geração da classificação final.

4.4 Classificadores Individuais do SISCOM

Uma das etapas para construção de um MCS é a criação dos Classificadores Individuais que irão compor o *"Pool de classificadores"* desse sistema. Em específico com a problemática dessa pesquisa de reconhecimento e classificação de emoções humanas a partir de imagens digitais, serão utilizados algoritmos CNNs de Aprendizado Profundo mais indicados para realização desse processo.

Um dos objetivos estabelecidos dessa dissertação foi de realizar um processo de finetuning utilizando os algoritmos de CNNs, a fim de se determinar a melhor arquitetura e metaparâmetros para essa rede proposta. Assim, dessa etapa serão selecionados os Classificadores Individuais que irão compor a solução SISCOM analisando o desempenho dos modelos e selecionados os que obtiverem melhor performance em relação as métricas geradas. A Figura 4.2 a seguir, traz um esquemático desse processo de comparação e seleção dos modelos.

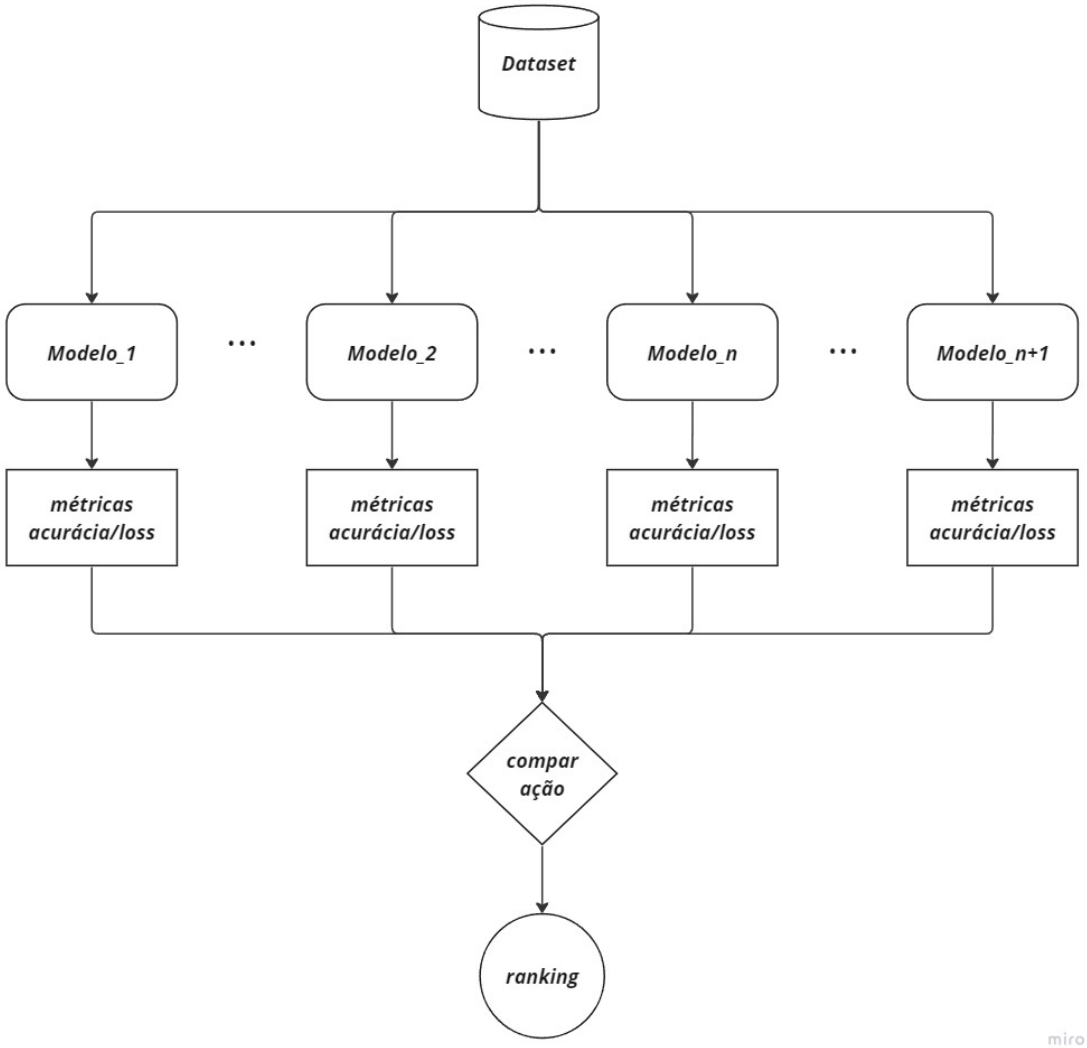

Figura 4.2: Esquemático processo de comparação e seleção Classificadores Individuais
Fonte: Autor

4.4.1 Arquitetura base

Pensando no desenvolvimento dos modelos dos Classificadores Individuais é estabelecido uma arquitetura base para a rede proposta. Dessa forma, todos os modelos implementados seguirão essa base, mas com variações de diferentes metaparâmetros durante o processo de finetuninng.

Nesse momento será detalhado a arquitetura base que os modelos seguirão para desenvolvimento

da rede proposta, para auxiliar o entendimento a Figura 4.3 trás o esquemático dessa arquitetura base.

Seguindo a referência da Figura 4.3, o primeiro passo é o redimensionamento das imagens para uma padronização de tamanho *(64X64)* pixels. Em seguida, essa imagem digital será a entrada para a primeira camada de *convolutional layer* que irá receber essa entrada e aplicar os filtros com a operação de convolução, por último será aplicado também o método de *"maxpooling"* a essa entrada.

Esse processo é repetido em sequência pelas duas outras camadas de convolutional layers responsáveis pela extração das características da imagem. Após as camadas de convolutional layers, a entrada também será processada por uma Rede Neural Profunda ou *"Deep Neural Network"* (DNN) que será responsável pela classificação das imagens a partir da extração das características da imagem realizada pelas convolutional layers anteriores.

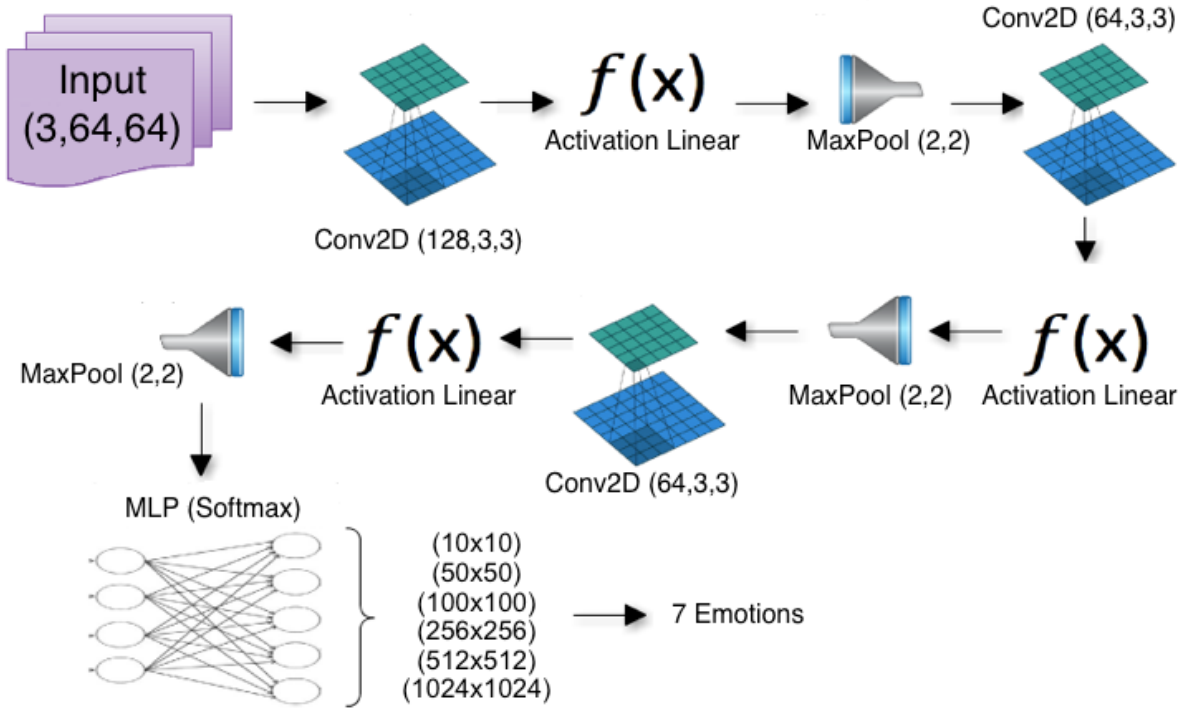

Figura 4.3: Arquitetura base dos modelos de Classificadores Individuais

Fonte: *Exploiting Convolutional Neural Networks to Recognize User's Emotion - Geraldo P. Rocha Filho, Vinícius P. Gonçalves*

4.4.2 Processo de finetunning

O denominado processo fine-tuning ou finetunning é uma etapa de extrema relevância, visando alcançar uma boa performance da rede proposta. Na literatura também denominado como aprendizado por transferência, essa técnica permite aproveitar o conhecimento prévio desenvolvido pelo modelo e adaptá-lo para tarefas específicas, economizando tempo e recursos computacionais.

Essa técnica inicia-se com o modelo base como ponto de partida e envolve o ajuste fino de

diversos metaparâmetros impactantes no desempenho dessa rede proposta. Assim o processo de finetuning visa adaptar essas características da melhor forma possível visando a tarefa específica da aplicação do modelo base, aproveitando o conhecimento prévio desse modelo e consequentemente gerando uma melhora no desempenho.

Os metaparâmetros são as propriedades de uma rede que devem ser testados, avaliados e selecionados as melhores atribuições possíveis com objetivo de atingir os melhores resultados. Dessa forma, será realizado o processo de finetunning ao longo das próximas seções dessa dissertação com objetivo de realizar o ajuste fino da arquitetura base já apresentada nos tópicos anteriores dos modelos de Classificadores Individuais, aplicando-se o método científico de teste e avaliação dos diversos metaparâmetros da rede.

Em relação ao método científico, serão testados e avaliados os seguintes metaparâmetros cruciais para um bom desempenho dos modelos de Classificadores Individuais:

- ***"Learning Rate"*** é um metaparâmetro que determina a ordem de grandeza em que os pesos atribuídos aos nós da rede serão atualizados, a Learning Rate tradicionalmente é definida entre um intervalo conforme apresentado na Equação 4.2.

$$0.00001 \leqslant LearningRate \leqslant 0.001 \tag{4.2}$$

  Ou seja, na prática ela define se a atualização dos pesos da rede será feita de uma forma menos intensa como no caso de uma *Learning Rate = (0.00001)* ou mais intensa no caso de uma *Learning Rate = (0.001)*.

- ***"Optmizer function"*** é um metaparâmetro que seu funcionamento determina o modo como os pesos atribuídos aos nós da rede serão atualizados em função do cálculo da métrica de perda determinado pela *"Loss function"*. Devem ser citados como tipos de Optmizer function os algoritmos de *Stochastic Gradient Descent (SGD)*, *Root Mean Square (RMSProp)* e *Adaptative Moment Estimation (ADAM)*;

- ***"Neurons"*** ou neurônios é um metaparâmetro fundamental para um modelo de Aprendizado de Máquina Profundo. São unidades computacionais que recebem um conjunto de entradas, realizam cálculos ponderados e aplicam uma função de ativação para produzir uma saída. A estrutura dos neurônios é inspirada no funcionamento dos neurônios biológicos do cérebro humano.

  Cada neurônio em uma camada de uma rede neural está conectado a todos os neurônios da camada anterior. E são essas conexões denominadas pesos da rede, que são ajustados durante o processo de treinamento para que o modelo aprenda a mapear corretamente os padrões nos dados de entrada para as saídas desejadas.

  Uma variação do número de neurônios em uma camada da rede afeta diretamente a capacidade de representação, a complexidade computacional, a regularização e a eficiência de treinamento do modelo. Por tanto, a escolha do número ideal de neurônios deve ser ajustada especificamente para cada modelo de Aprendizado de Máquina Profundo;

- ***"Convolutional Layer"*** é um componente crucial em modelos de Aprendizado de Máquina Profundo, especialmente para CNNs, essa camada é responsável por realizar operações de convolução em dados de entrada, visando extrair características relevantes. A variação do número de camadas de convolução em um modelo de Aprendizado de Máquina Profundo pode afetar significativamente o funcionamento e desempenho do modelo.

  Aumentar o número de camadas de convolução permite que o modelo aprenda representações mais complexas e hierárquicas das características presentes nos dados de entrada. Isso é importante para lidar com problemas mais desafiadores, onde as informações relevantes estão distribuídas em várias escalas e níveis de abstração. A escolha adequada do número de camadas de convolução depende do problema em questão, da disponibilidade de dados de treinamento e dos recursos computacionais disponíveis, assim, esse será o metaparâmetro avaliado;

4.4.3 Fase de testes

Nessa subseção será detalhado o método científico que foi aplicado para realizar os testes e avaliações do processo de finetunning, seguindo a metodologia desenvolvida no artigo científico *"Exploiting Convolutional Neural Networks to Recognize User's Emotion"* dos autores *Geraldo P. Rocha Filho e Vinícius P. Gonçalves* (14). O método aplicado consiste em uma divisão em três etapas de fase de testes distintos, visando implementar e configurar os modelos de Classificadores Individuais a partir da arquitetura base já apresentada anteriormente e realizar variações nos metaparâmetros específicos citados e analisar a performance de cada variação.

O método científico resume-se então em realizar a divisão em três etapas de fases de testes, em que cada etapa será focada na análise de um metaparâmetro específico e suas variações. Por tanto, define-se as fases de testes conforme:

- ***"Fase de testes I"*** será avaliado o metaparamêtro de *"Optimizer function"*, variando-se os algortimos de *(SGD, ADAM e RMSPROP)*. Nessa mesma fase também será avaliado o metaparamêtro de *"Learning rate"* variando-se com os possíveis valores de *(0.001, 0.0001 e 0.00001)*, no total serão realizadas nove execuções com todas as variações descritas e avaliado os resultados;

- ***"Fase de testes II"*** será avaliado o metaparamêtro de número de *"Neurons"* da *Rede Neural Profunda (DNN)* ao fim da extração de informações das imagens nas camadas de convolutional layers, variando-se então com os possíveis valores de *(10, 50, 100, 256, 512 e 1024)*. A proposta dessas variações é verificar uma possível redução nas configurações da rede DNN, no total serão realizadas seis execuções com todas as variações descritas e avaliado os resultados;

- ***"Fase de testes III"*** nessa útlima etapa serão utilizados os melhores resultados obtidos das execuções das duas fases de testes anteriores. Com esses resultados dos melhores metaparâmetros de *(Optmizer function, Learning Rate e Neurons)* já determinados e selecionados, a

proposta é utiliza-los na implementação da melhor arquitetura da Rede Neural Convolucional (CNN). E o metaparâmetro que será variado e analisado é o *"Número de camadas de Convolutional Layers"* dessa CNN, variando-se com os possíveis valores *(2, 3, 4 e 5)*, no total serão realizadas quatro execuções com todas as variações descritas e avaliados os resultados;

Em conclusão, ao final da última etapa de testes espera-se ter alcançado os objetivos específicos dessa dissertação de realizar o processo de finetuning da rede proposta utilizando algoritmos de CNNs e gerado os Classificadores Individuais a partir dos melhores desempenhos observados. Com os resultados obtidos será então possível realizar a seleção dos melhores Classificadores Individuais para composição futura do Sistema Comitê SISCOM.

Capítulo 5

Metodologia experimental

Este capítulo detalha de forma prática os procedimentos experimentais da pesquisa, a implementação e desenvolvimento dos modelos de Classificadores Individuais, Sistema de Comitê SISCOM e a consolidação do Dataset utilizado nos experimentos.

5.1 Dataset

Na construção de um modelo de Aprendizado de Máquina, a fase de Coleta de Dados (Data Collection) é de extrema importância e representa uma das etapas iniciais fundamentais. A qualidade dos dados utilizados tem um impacto direto no desempenho e na eficácia do modelo, pois dados defeituosos ou imprecisos podem resultar em um modelo insatisfatório que não reflete a precisão desejada nos resultados.

Portanto, a coleta de dados é um processo minucioso, exigindo esforço significativo para garantir que os dados sejam confiáveis e relevantes para a tarefa em questão. A identificação das fontes de dados adequadas é crucial nessa etapa. Isso envolve a seleção criteriosa das melhores fontes de dados que serão utilizadas para compor o conjunto de dados (dataset). A preparação dos dados também ocorre nessa etapa, garantindo que os dados utilizados são íntegros, governados e seguros como no caso da utilização de imagens digitas que deve-se garantir que nehum dos arquivos incluídos no dataset possa estar corrompido.

5.1.1 Base de dados

Com base na compreensão de todos os passos envolvidos, o projeto fez uso do dataset original do artigo intitulado "Exploiting Convolutional Neural Networks to Recognize User's Emotion"(14). Esse conjunto de dados foi compilado a partir de várias bases de dados de imagens digitais que representam expressões faciais, abrangendo as emoções humanas básicas. Os conjuntos de dados utilizados na composição foram provenientes das seguintes fontes: RaFD databases Radboud Faces (29), CK+ Extended Cohn-Kanade (31), IMPA-FACE3D (20) e FACES (27). Importante ressaltar que todas essas bases de dados estão disponíveis de forma gratuita.

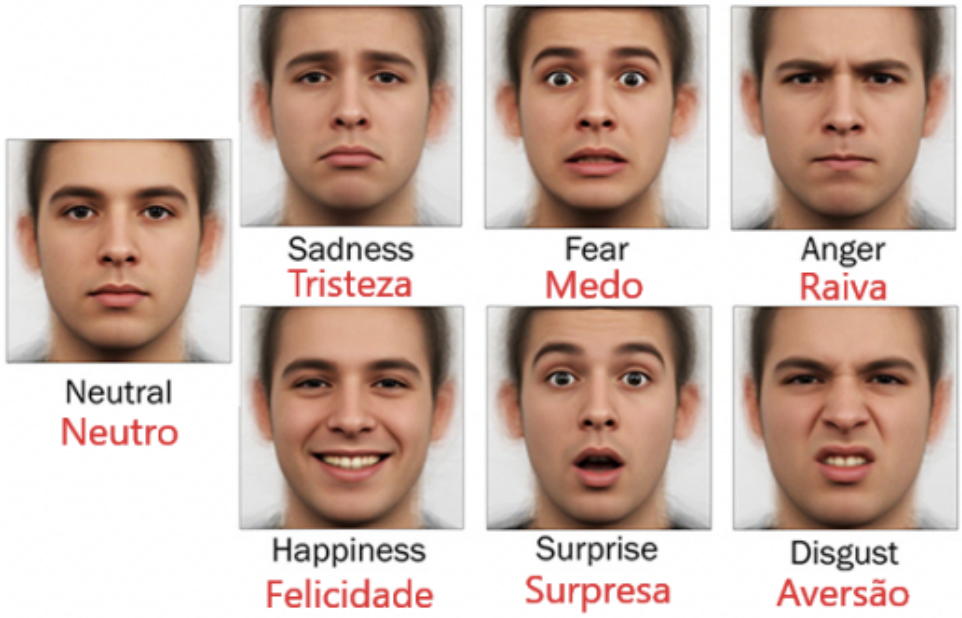

Figura 5.1: Representação das emoções humanas básicas

Ao combinar essas fontes, o dataset resultante abrangeu uma ampla variedade de expressões faciais humanas, o que contribuiu para aumentar a riqueza e diversidade dos dados utilizados no projeto. A inclusão de múltiplas bases de dados também permitiu abranger diferentes contextos e características emocionais, proporcionando uma representação mais abrangente do espectro emocional humano.

A disponibilidade gratuita dessas bases de dados também facilitou a acessibilidade ao conjunto de dados composto, permitindo que outros pesquisadores e profissionais na área possam utilizá-lo em suas próprias investigações e experimentos relacionados a reconhecimento de emoções e análise facial.

5.1.2 Classificação das expressões

Na seleção das diferentes bases de dados, todas as imagens que representam as emoções básicas humanas, incluindo Aversão, Felicidade, Medo, Neutro, Raiva, Surpresa e Tristeza, foram cuidadosamente escolhidas. Essa abordagem garantiu que o dataset composto abrangesse um espectro completo de emoções faciais, tornando-o representativo e diversificado.

Ao compor o dataset, foi dada especial atenção à inclusão de imagens que refletissem uma variedade de testes heterogêneos. Isso significa que as imagens foram coletadas de pessoas pertencentes a diferentes regiões geográficas, culturas e etnias. Essa diversidade cultural foi propositalmente incorporada ao conjunto de dados para que o modelo de Aprendizado de Máquina desenvolvido fosse aplicável em qualquer dispositivo computacional, independentemente das características culturais, étnicas ou costumes do usuário.

5.1.3 Consolidação do dataset

A consolidação final do dataset resultou em um total de 3115 imagens. Essas imagens foram divididas de forma equilibrada em 445 imagens para cada uma das emoções representadas. Esse cuidadoso balanceamento garante que o conjunto de dados seja ideal para a realização de testes e análises confiáveis.

A Tabela 5.1 apresenta a composição detalhada do dataset, mostrando o número de imagens extraídas de cada uma das diferentes bases de dados. O objetivo principal foi obter um balanceamento correto para o número de imagens associadas a cada emoção específica.

Esse balanceamento é de extrema importância para garantir que o modelo seja treinado e testado de forma justa e equilibrada em relação a todas as emoções humanas representadas. Dessa forma, cada emoção tem uma quantidade igual de exemplos para o treinamento e teste do modelo, evitando assim qualquer viés ou desequilíbrio que poderia afetar a performance e a generalização do modelo.

Tabela 5.1: Composição do Dataset por Emoção

Emoção	**Radboud Faces**	**CK+**	**IMPA-FACE3D**	**FACES**
Aversão	67	169	38	171
Felicidade	67	169	38	171
Medo	67	169	38	171
Neutro	67	169	38	171
Raiva	67	169	38	171
Surpresa	67	169	38	171
Tristeza	67	169	38	171

O dataset consolidado utilizado para todos os experimentos desse trabalho, está acessível na plataforma Github atráves do link: https://github.com/lp-mateus/UNB_TCC_SISCOM_DATASET. Disponível para consulta e utilização em futuras pesquisas científicas na área.

5.2 Classificadores Individuais SISCOM

Nesse momento será apresentado o desenvolvimento e implementação dos modelos de Classificadores Individuais, seguindo o método científico detalhado anteriormente para o processo de finetunning.

Foram utilizados para tal implementação a ferramenta do Google Colab fornecendo a infraestrutura de hardware necessário para treinamento dos modelos, a máquina alocada do Google Colab Notebook possuía as especificações de hardware: *(CPU: Intel(R) Xeon(R) CPU @ 2.30GHz), (GPU: NVIDIA Tesla K80 12GB GDDR5 VRAM), (RAM: 12GB).*

Já para desenvolvimento do código foi utilizado a linguagem *Python* e a *API Keras* de Aprendizado de Máquina Profundo escrita em Python e executada pela plataforma de aprendizado de

máquina TensorFlow. A *"API Keras Deep Learning"* é extremamente robusta e poderosa fornecendo desempenho e escalabilidade, realiza uma execução eficiente de operações de tensor de baixo nível na (CPU, GPU ou TPU), calcula o gradiente de expressões diferenciáveis arbitrárias, realiza dimensionamento da computação para muitos dispositivos, exportação de programas gráficos para ambientes de execução externos, como servidores, navegadores e dispositivos móveis.

5.2.1 Fase de testes I

Seguindo o método científico descrito para realização da Fase de testes I, o foco dessa etapa é a avaliação dos metaparâmetros de Optimezer function e Learning Rate. Dessa forma, seguindo a arquitetura base para o modelo dos Classificadores Individuais ilustrado na Figura 4.3 serão desenvolvidos e implementados modelos com todas as variações desses metaparâmetros. A Tabela 5.2 a seguir resume as variações desses modelos:

Tabela 5.2: Metaparâmetros avaliados na Fase de testes I

Optmizer function	***Learning Rate***
SGD	0.001
SGD	0.0001
SGD	0.00001
ADAM	0.001
ADAM	0.0001
ADAM	0.00001
RMSPROP	0.001
RMSPROP	0.0001
RMSPROP	0.00001

Serão realizadas nove execuções nessa Fase de testes I, para cada variação descrita em que os modelos seguirão a seguinte configuração *CONFIG = CNN(3) - DNN(128,128,7)*. O trecho de código 5.1 apresentado a seguir, detalha a implementação dos modelos sendo possível visualizar as três camadas iniciais de Convolutional layers, sendo a primeira de entrada que recebe a imagem digital com tamanho *(64X64)* pixels.

Em sequência ocorre aplicação dos filtros com a operação de convolução e método de *"ReLU"* como Activation function e pelo método de *"Maxpool"*, após a extração das características segue para o processamento das três camadas da rede DNN que irá realizar a classificação da entrada a partir das informações extraídas anteriormente. Os metaparâmetros Optimizer function e Learning Rate são definidas nas linhas de código 22-23 apresentada nas configurações de compilação do modelo e nesse trecho serão realizadas as variações dos valores descritos.

```
1 model = Sequential()
2
3 # CNN LAYER 1
4 model.add(Conv2D(input_shape=(64,64,3),filters=128,kernel_size=(3,3),padding="
      same", activation="relu"))
```

```
5  model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
6  # CNN LAYER 2
7  model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"
       ))
8  model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
9  # CNN LAYER 3
10 model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"
       ))
11 model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
12
13 model.add(Flatten())
14
15 # DNN LAYER 1
16 model.add(Dense(128, activation='relu'))
17 # DNN LAYER 2
18 model.add(Dense(128, activation='relu'))
19 # DNN LAYER 3
20 model.add(Dense(7, activation='softmax'))
21
22 # OPT AND LR
23 model.compile(optimizer=SGD(learning_rate=0.001),loss='categorical_crossentropy',
       metrics=['accuracy'])
24
25 model.summary()
```

Trecho de código 5.1: Implementação modelo CNN para Fase de testes I

5.2.2 Fase de testes II

Já para realização da Fase de testes II, o foco dessa etapa é a avaliação do metaparâmetro de Neurons da Rede Neural Profunda (DNN) ao fim da extração de informações das imagens nas camadas de Convolutional layers. Dessa forma, seguindo a arquitetura base para o modelo dos Classificadores Individuais detalhado na Figura 4.3 serão desenvolvidos e implementados modelos com todas as variações desse metaparâmetro. A Tabela a seguir 5.3 resume as variações desses modelos:

Tabela 5.3: Metaparâmetros avaliados na Fase de testes II

Neurons	10	50	100	256	512	1024

Serão realizadas seis execuções nessa Fase de testes II, para cada variação descrita em que os modelos seguirão a seguinte configuração *CONFIG = CNN(3) - DNNN(X,7)* em que *X* é a variação dos números de Neurons da DNN que serão testados. O trecho de código 5.2 apresentado a seguir, detalha a implementação dos modelos, em que a mesma estrutura foi utilizada para as camadas de Convolutional layers apresentadas anteriormente mas a alteração ocorre na rede DNN, utilizando-se apenas duas camadas. Destaque-se a linha 19 do trecho de código na qual é implementada a camada de *"Dense layer"* onde é variado o número de Neurons dessa mesma camada.

```
model = Sequential()

# CNN LAYER 1
model.add(Conv2D(input_shape=(64,64,3),filters=128,kernel_size=(3,3),padding="
    same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.5))
# CNN LAYER 2
model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"
    ))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.5))
# CNN LAYER 3
model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"
    ))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.5))

model.add(Flatten())

# DNN LAYER 1
model.add(Dense(10, activation='relu'))
model.add(Dropout(0.5))
# DNN LAYER 2
model.add(Dense(7, activation='softmax'))

# OPT AND LR
model.compile(optimizer=Adam(learning_rate=0.0001),loss='categorical_crossentropy
    ',metrics=['accuracy'])

model.summary()
```

Trecho de código 5.2: Implementação modelo CNN para Fase de testes II

5.2.3 Fase de testes III

Para a realização da última etapa Fase de testes III, serão utilizados os melhores resultados obtidos das execuções das duas fases de testes anteriores com os resultados dos melhores metaparâmetros de (Optmizer function, Learning Rate e Neurons) já determinados e selecionados. O foco dessa etapa é a avaliação do metaparâmetro de (Número de camadas de Convolutional Layers) da da Rede Neural Convolucional (CNN). Dessa forma, seguindo a arquitetura base para o modelo dos Classificadores Individuais detalhado na Figura 4.3 serão desenvolvidos e implementados modelos com todas as variações desse metaparâmetro. A Tabela 5.4 a seguir resume as variações desses modelos:

Tabela 5.4: Metaparâmetros avaliados na Fase de testes III

Convolutional Layers	2	3	4	5

Serão realizadas quatro execuções nessa Fase de testes III, para cada variação descrita em que os modelos seguirão a seguinte configuração *CONFIG = CNN(X) - DNN(256,256,7)* em que *X* é variação do número de camadas de Convolutional layers que serão testados. O trecho de código 5.3 apresentado a seguir, detalha a implementação dos modelos, em que a mesma estrutura foi utilizada para as camadas de Convolutional layers apresentadas anteriormente mas com a variação do número de camadas. Destaquem-se as linhas 3-10 explicitando-se a variação do número de camadas aplicadas para rede CNN.

```
1  model = Sequential()
2
3  # CNN LAYER 1
4  model.add(Conv2D(input_shape=(64,64,3),filters=128,kernel_size=(3,3),padding="
       same", activation="relu"))
5  model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
6  model.add(Dropout(0.5))
7  # CNN LAYER 2
8  model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"
       ))
9  model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
10 model.add(Dropout(0.5))
11
12 model.add(Flatten())
13
14 # DNN LAYER 1
15 model.add(Dense(256, activation='relu'))
16 model.add(Dropout(0.5))
17 # DNN LAYER 2
18 model.add(Dense(256, activation='relu'))
19 model.add(Dropout(0.5))
20 # DNN LAYER 3
21 model.add(Dense(7, activation='softmax'))
22
23 # OPT AND LR
24 model.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy'
       ,metrics=['accuracy'])
25
26 model.summary()
```

Trecho de código 5.3: Implementação modelo CNN para Fase de testes III

5.2.4 Treinamento dos modelos

Uma vez implementado o código referente as redes CNNs das *(Fases de testes I, II e III)* e suas respectivas variações contemplando todos metaparâmetros que serão testados nas diferentes execuções, o próximo passo em relação a implementação dos Classificadores Individuais foi a realização do treinamento dos modelos para cada execução.

Para o treinamento desses modelos adotou-se um padrão para todas as execuções, visando estabelecer as mesmas condições de treinamento a fim de posteriormente avaliar os resultados e

estabelecer conclusões baseadas nos padrões adotados.

O treco de código a seguir 5.4 detalha o método *"model.fit"* responsável por realizar o treinamento do modelo iterando sobre os dados de entrada, calculando os gradientes e atualizando os pesos do modelo de acordo com o algoritmo de otimização especificado na compilação do modelo. O objetivo é minimizar a função de perda definida durante a compilação do modelo, esse método retorna um objeto *"history"* que contém informações sobre as métricas de treinamento e validação durante as épocas que será utilizado para analisar o desempenho do modelo.

```
# Treinamento do modelo
history = model.fit(training_set,batch_size=BATCH_SIZE,epochs=500,verbose=1,
    validation_data=test_set,shuffle=True)
```

Trecho de código 5.4: Implementação do treinamento padronizado para os modelos de CNNs

A tabela 5.5 a seguir resume as propriedades padrões utilizadas em todas as execuções dos treinamentos auxiliando na compreensão do código implementado:

Tabela 5.5: Configuração padrão do treinamento dos modelos de CNNs

Epochs	***Batch Size***	***Verbose***
500	64	1

5.2.5 Métricas de avaliação

Inicialmente detalha-se as métricas de avaliação que serão geradas, em relação aos treinamentos para cada modelo de Classificador Individual durante as Fases de testes do processo de finetunning. As métricas de avaliação servirão como base para análise e comparação dos resultados e seleção dos melhores modelos de Classificadores Individuais.

O trecho de código 5.5 a seguir, detalha o método *"model.compile()"* da API Keras que realiza a configuração do processo de treinamento do modelo. Essa função é utilizada para especificar a função de perda *"loss"* que será otimizada durante o treinamento e as métricas que serão utilizadas para avaliar o desempenho do modelo.

```
# Metricas do treinamento do modelo
model.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy'
    ,metrics=['accuracy'])
```

Trecho de código 5.5: Configuração das métricas utilizadas para treinamento dos modelos

Em específico para o treinamento dos modelos de Classificadores Individuais optou-se pela loss como *"categorical_ crossentropy"* comumente utilizada em problemas de classificação multiclasse, onde cada amostra pode pertencer a apenas uma classe. Essa função de perda é apropriada quando as classes são mutuamente exclusivas, ou seja, cada amostra pertence a apenas uma classe. Além disso, as previsões do modelo são normalmente passadas por uma camada de ativação softmax para converter as saídas do modelo em probabilidades, permitindo a comparação com os rótulos verdadeiros.

Destaca-se também a configuração da métrica de *"acurácia"*, comumente utilizada em problemas de classificação. Essa métrica representa a proporção de amostras classificadas corretamente pelo modelo em relação ao total de amostras. A escolha da acurácia como métrica é útil para entender o desempenho geral do modelo.

Em sequência com o modelo já treinado, é possível verificar as métricas (acurácia e loss) citadas a partir do método *"model.history()"* da API Keras. O atributo *"history"* é um dicionário que contém métricas de treinamento que são retornadas após a conclusão do mesmo. Esse atributo também contém informações sobre a evolução da perda durante o processo de treinamento, o trecho de código 5.6 a seguir destaca essa implementação.

```
# Gerando metricas acc e loss do treinamento
view_train_loss = history.history['loss']
view_test_loss = history.history['val_loss']

view_train_accuracy = history.history['accuracy']
view_test_accuracy = history.history['val_accuracy']
```

Trecho de código 5.6: Geração das métricas de acurácia e loss do treinamento

Em seguida é realizado a etapa de Visualização dos Dados *"Data Visualization"*, a partir da geração dos gráficos das métricas de (acurácia e loss) ao longo do treinamento, utilizando o atributo *"history"*. O trecho de código 5.7 a seguir detalha a geração dos gráficos para apoio a análise dos resultados.

```
# Plotting a line chart to visualize the loss and accuracy values by epochs
fig, ax = plt.subplots(ncols=2, figsize=(20,10))

ax = ax.ravel()

# Loss
ax[0].plot(view_train_loss, label='Train Loss', color='royalblue')
ax[0].plot(view_test_loss, label='Test Loss', color = 'orangered')

ax[0].set_xlabel('Epochs', fontsize=14)
ax[0].set_ylabel('Categorical Crossentropy', fontsize=14)

ax[0].legend(fontsize=14)
ax[0].tick_params(axis='both', labelsize=12)

# Accuracy
ax[1].plot(view_train_accuracy, label='Train Accuracy', color='royalblue')
ax[1].plot(view_test_accuracy, label='Test Accuracy', color='orangered')

ax[1].set_xlabel('Epochs', fontsize=14)
ax[1].set_ylabel('Accuracy', fontsize=14)

ax[1].legend(fontsize=14)
ax[1].tick_params(axis='both', labelsize=12)

# Captions
```

```
fig.suptitle(x=0.5, y=0.95, t="Metrics loss and accuracy of CNN model by epochs",
    fontsize=16)

# Saving the figure
plt.savefig('output_metrics.png')
```

Trecho de código 5.7: Código de implementação da etapa de Data Visualization

O último passo é a obtenção consolidada das métricas de (acurácia e loss) citadas através do método "model.evaluate()"da API Keras, utilizado para avaliar o desempenho do modelo treinado em relação ao dataset de teste. Esse método calcula a loss e quaisquer métricas adicionais especificadas durante a compilação do modelo. O trecho de código 5.8 a seguir detalha essa implementação final.

```
# Avaliando metricas consolidadas do modelo com o dataset "test"
model.evaluate(test_set)
```

Trecho de código 5.8: Obtenção das métricas consolidadas do treinamento

5.3 Sistema Comitê SISCOM

Nesse momento será apresentado o desenvolvimento e implementação da solução proposta Sistema Comitê SISCOM, como detalhado nas seções do Capítulo 4 dessa dissertação a opção pela divisão em módulos da arquitetura deu-se também para facilitar a implementação dessa solução conforme já ilustrado na Figura 4.1.

Conforme estabelecido pelo método científico descrito para construção da solução SISCOM, o objetivo é utilizar os Classificadores Individuais de melhores performances selecionados anteriormente no processo de finetunning para composição do denominado Pool de classificadores.

Dessa forma, serão desenvolvidos e integrados os seguintes módulos (Módulo dataset, Módulo pool de classificadores, Módulo de decisão) e nas subseções a seguir serão detalhadas as implementações em código de cada módulo citado.

Novamente foi utilizado para implementação a ferramenta do Google Colab fornecendo a infraestrutura de hardware necessário para treinamento dos modelos, a máquina alocada do Google Colab Notebook possuía as especificações de hardware: *(CPU: Intel(R) Xeon(R) CPU @ 2.30GHz), (GPU: NVIDIA Tesla K80 12GB GDDR5 VRAM), (RAM: 12GB).* E para o desenvolvimento do código novamente foi utilizado a linguagem *Python* e a *API Keras* de Aprendizado de Máquina Profundo escrita em Python e executada pela plataforma de aprendizado de máquina TensorFlow.

5.3.1 Módulo dataset

Para o desenvolvimento do Módulo Dataset, foram importadas três bibliotecas essenciais: *os*, *glob* e *matplotlib.pyplot*. Com o uso dessas bibliotecas, é possível importar, manipular, formatar e visualizar as imagens que serão utilizadas para criar os modelos de análise de dados ou aprendizado

de máquina. Essas etapas são fundamentais para garantir que o dataset esteja pronto para ser utilizado com eficácia e precisão nos modelos de análise ou treinamento.

```
# Bibliotecas padr o
import matplotlib.pyplot as plt
import glob as gb
import os
```

Trecho de código 5.9: Importação de bibliotecas padrão

No Google Colab, para acessar e importar as imagens armazenadas no Google Drive, é necessário utilizar a biblioteca *google.colab* que permite interagir com o Google Drive e obter as permissões necessárias para acessar as pastas e os arquivos. As variáveis configuradas em seguida são os caminhos completos para o diretório com as imagens de treinamento e teste, respectivamente.

```
from google.colab import drive
drive.mount('/content/drive')
TRAIN_DIR = '/content/drive/MyDrive/ConsolidadoPaperVersion_Dataset/train'
TEST_DIR = '/content/drive/MyDrive/ConsolidadoPaperVersion_Dataset/test'
```

Trecho de código 5.10: Importação da biblioteca do Drive

O código 5.11 usa a biblioteca *tensorflow.keras.utils* para carregar imagens de um diretório específico e exibi-las usando a biblioteca *matplotlib.pyplot*. Exibe uma grade de 9 imagens da classe 'happy' (feliz) carregadas do diretório de treinamento. Cada imagem é redimensionada para um tamanho de (256, 256) pixels e é exibida em uma célula da grade. Esse tipo de visualização é útil para visualizar algumas amostras do conjunto de dados de treinamento e verificar a qualidade das imagens antes de treinar um modelo de aprendizado de máquina. O retorno desse bloco de código pode ser visto na figura 5.2.

```
from tensorflow.keras.utils import load_img, img_to_array

expression = 'happy'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
    img = load_img(TRAIN_DIR + "/" + expression + "/" +
                   os.listdir(TRAIN_DIR + "/" + expression)[i], target_size
    =(256,256))
    plt.imshow(img)
plt.show()
```

Trecho de código 5.11: Verificação das Imagens

No código 5.12, são criados dois objetos *ImageDataGenerator*: *train_datagen* e *test_datagen*. Esses objetos são utilizados para realizar a pré-processamento das imagens antes de alimentá-las no modelo de aprendizado. São usados dois parâmetros para a criação desses objetos: *rescale* e *horizontal_flip*.

Neste código, o *rescale* está configurado para dividir os valores dos pixels por 255 para normalizar os valores de pixel entre 0 e 1. Essa etapa é importante, pois a maioria dos modelos de

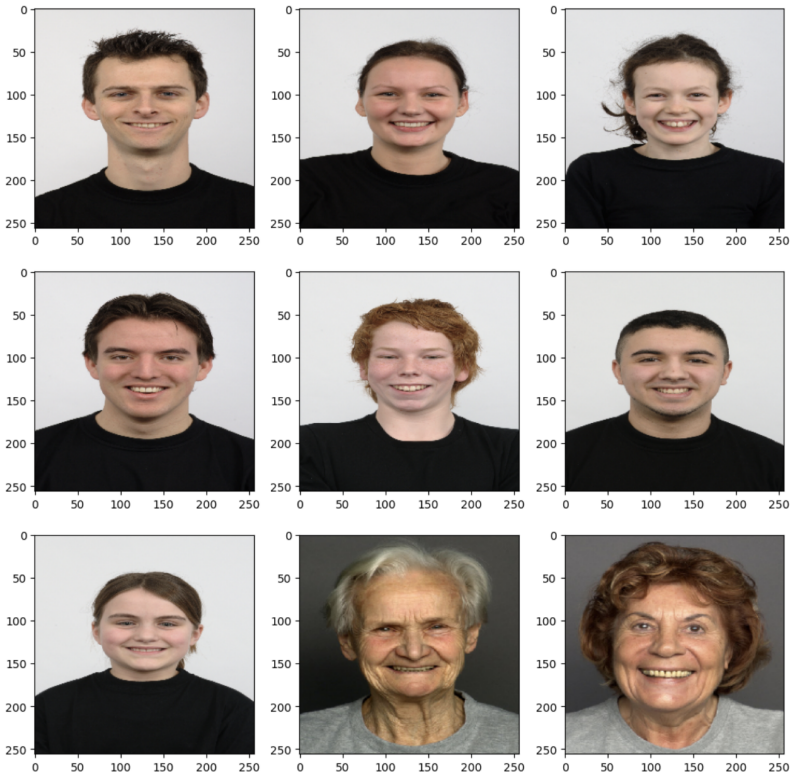

Figura 5.2: Visualização de amostra do dataset de treinamento

Fonte: Autor

aprendizado de máquina funciona melhor com valores de entrada normalizados. Já o *horizontal_flip* está configurado como **true** para o *train_datagen* e **false** para o *test_datagen*. Isso ocorre porque esse parâmetro é responsável por realizar a operação de espelhamento horizontal aleatório nas imagens durante o treinamento. Essa técnica é útil para aumentar o tamanho do conjunto de dados e melhorar a capacidade de generalização do modelo. Porém, o espelhamento horizontal não é aplicado ao conjunto de teste, pois não é desejado que as imagens sejam modificadas durante a avaliação do modelo.

```
BATCH_SIZE = 64
train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

Trecho de código 5.12: Geração do Dataset

O objeto *training_set* é criado usando o método *flow_from_directory* do *train_datagen*. Esse método carrega as imagens do diretório especificado (*TRAIN_DIR*) e gera os lotes (batches) de dados para o treinamento do modelo. O objeto *test_set* é criado da mesma maneira que o *training_set*, mas usando o *test_datagen* e *TEST_DIR*. Esse conjunto de dados é usado para avaliar o desempenho do modelo após o treinamento.

```
training_set = train_datagen.flow_from_directory(
                    TRAIN_DIR,
                    target_size = (64, 64),
```

```
                    batch_size = BATCH_SIZE,
                    color_mode = "rgb",
                    shuffle=True,
                    class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(
                    TEST_DIR,
                    target_size = (64, 64),
                    batch_size = BATCH_SIZE,
                    color_mode = "rgb",
                    shuffle=True,
                    class_mode = 'categorical')
```

Trecho de código 5.13: Geração do Dataset

5.3.2 Módulo pool de classificadores

Nessa etapa para desenvolvimento do Módulo Pool de classificadores da solução SISCOM, foi realizado uma integração com os modelos Classificadores Individuais selecionados anteriormente no processo de finetunning. A estratégia aqui utilizada foi após a execução de cada treinamento dos modelos de Classificadores individuais foi aplicado o método *"model.save"* uma função específica do módulo *"keras.models"* que permite salvar um modelo treinado em disco.

Esse método utiliza o formato de arquivo *Hierarchical Data Format* (HDF5) para armazenar o modelo, incluindo sua arquitetura, pesos, configuração e até mesmo o otimizador utilizado durante o treinamento. O trecho de código 5.14 a seguir detalha essa implmentação:

```
# Salvando modelo
model.save('modelo_ft02_exc04.h5')
```

Trecho de código 5.14: Método utilizado para armazenar em disco os modelos treinados

Em posse dos modelos treinados, o passo seguinte é realizar o carregamento dos diferentes conjuntos de Classificadores Individuais selecionados para composição do Pool de classificadores. O método *"load_model"* da API Keras nos permite carregar um modelo previamente treinado, salvo em disco, possibilitando a integração desses diferentes modelos de Classificadores Individuais. O trecho de código 5.15 a seguir detalha essa implementação:

```
from keras.models import load_model
from sklearn.metrics import accuracy_score
import numpy as np

# loading models
model1 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
    modelo_ft01_exc05.h5')
model2 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
    modelo_ft02_exc04.h5')
model3 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
    modelo_ft03_exc04.h5')
```

Trecho de código 5.15: Método utilizado para integração dos modelos treinados

5.3.3 Módulo de decisão

A etapa final para construção da solução SISCOM é o desenvolvimento do Módulo de decisão, nesse momento já realizamos a integração dos modelos Classificadores Individuais e o objetivo é realizar a combinação das predições desses modelos e obter uma decisão final com maior precisão.

Seguindo o método aplicado para desenvolvimento do módulo de decisão, optou-se pela escolha da média ponderada, de forma que as predições de cada classificador são ponderadas de acordo com sua confiabilidade determinada pela performance obtida individualmente. A estratégia aqui adotada foi de atribuir um maior peso, ou seja, uma maior responsabilidade no momento de decisão proporcional a performance desses modelos.

O trecho de código 5.16 a seguir, detalha a implementação do módulo de decisão. Inicialmente com os modelos treinados já integrados é possível realizar várias operações com os mesmos, utilizando-se da API Keras. Conforme explicitado nas linhas 1-4 foi utilizado o método *"model.predict()"*, uma funcionalidade fundamental fornecida pela API Keras para realizar previsões a partir de modelos já treinados.

```
1  # combine predictions
2  predictions1 = model1.predict(test_set)
3  predictions2 = model2.predict(test_set)
4  predictions3 = model3.predict(test_set)
5
6  # ensemble method weight average
7  weights = [0.5, 0.2, 0.3]
8  preds = np.array([predictions1, predictions2, predictions3])
9  weighted_preds = np.tensordot(preds, weights, axes=((0),(0)))
10 weighted_ensemble_prediction = np.argmax(weighted_preds, axis=1)
11
12 # calculate ensemble accuracy
13 ensemble_accuracy = accuracy_score(test_set.labels, weighted_ensemble_prediction)
14 print('Accuracy Score for average weight ensemble model = ', ensemble_accuracy)
```

Trecho de código 5.16: Código de implementação do Módulo de decisão

Esse método realiza inferências sobre os dados de entrada fornecidos e retorna uma matriz do tipo *"Numpy"* (ou uma lista de matrizes Numpy, se o modelo tiver várias saídas) contendo as previsões correspondentes ao dado de entrada fornecido.

Já nas linhas 6-10, detalha-se a implementação do método de combinação por média ponderada das predições obtidas por cada modelo. Destaca-se os pesos atribuídos na ordem dos respectivos modelos (*"model1", "model2", "model3"*), em seguida é consolidado em uma estrutura de dado array *"preds"*, o conjunto das predições de cada modelo e realizado a operação matemática da média ponderada. Por fim, o retorno dessa operação matemática é a estrutura de dado final denominada *"weighted_ensemble_prediction"* sendo justamente a combinação final que desejamos para a solução Sistema Comitê SISCOM.

Destaca-se as linhas finais 12-14 de obtenção da métrica de acurácia da solução SISCOM proposta, aqui é utilizado o método *"accuracy_score()"* da biblioteca de Aprendizado de Máquina

Scikit-learn . Esse método é uma métrica de avaliação muito comum em problemas de classificação, utilizada para avaliar a acurácia de um modelo, comparando as previsões feitas pelo modelo com as classes reais dos dados de teste.

Capítulo 6

Resultados

Este capítulo apresenta os resultados obtidos para as diversas execuções dos modelos de Classificadores Individuais, realizadas durante o processo de finetunning. Serão analisadas e comparadas as métricas de performances obtidas durante o treinamento desses modelos. E por fim, serão destacados e selecionados os melhores modelos de acordo com as performances obtidas para composição do Módulo Pool de Classificadores da solução SISCOM.

Em relação a solução SISCOM, também serão apresentados os resultados obtidos das duas configurações *(SISCOM I e SISCOM II)*. Serão analisadas as métricas de performance (Acurácia, Loss, CCP) em relação as duas configurações citadas.

Por último, ainda nesse capítulo com o intuito de comprovar o objetivo da proposta, métodos estatísticos foram utilizados para validar os resultados. Seguindo a teoria descrita no Capítulo 2 da Fundamentação Teórica dessa dissertação, serão apresentados os resultados das análises estatísticas aplicadas.

6.1 Classificadores Individuais SISCOM

Nessa subseção serão detalhados os resultados das Fases de testes, obtidos das execuções em relação aos treinamentos dos modelos de Classificadores Individuais. Dessa forma, na sequência dessa dissertação foi organizado a apresentação desses resultados conforme *(Fase de testes I, Fase de testes II, Fase de testes III)* e por cada execução dos treinamentos dos modelos. Também serão analisadas as métricas de performance *(Acurácia)* e *(Loss)*, além da métrica de *(Custo Computacional)* CCP, o tempo de treinamento de cada modelo que servirão de base para análise dos resultados.

6.1.1 Fase de testes I

Conforme descrito na Tabela 5.2, para realização da Fase de testes I serão realizadas nove execuções contemplando todas as variações de metaparâmetros que serão testados para o processo de finetunning. Ressaltando para essa fase de teste a configuração da arquitetura implementada:

CONFIG = CNN(3) - DNN(128,128,7).

6.1.1.1 Primeira execução

Em relação a primeira execução foi implementado a variação dos metaparâmetros *Optimizer = (SGD)* e *Learning_Rate = (0.001)*, para realização do treinamento desse modelo. A Figura 6.1 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

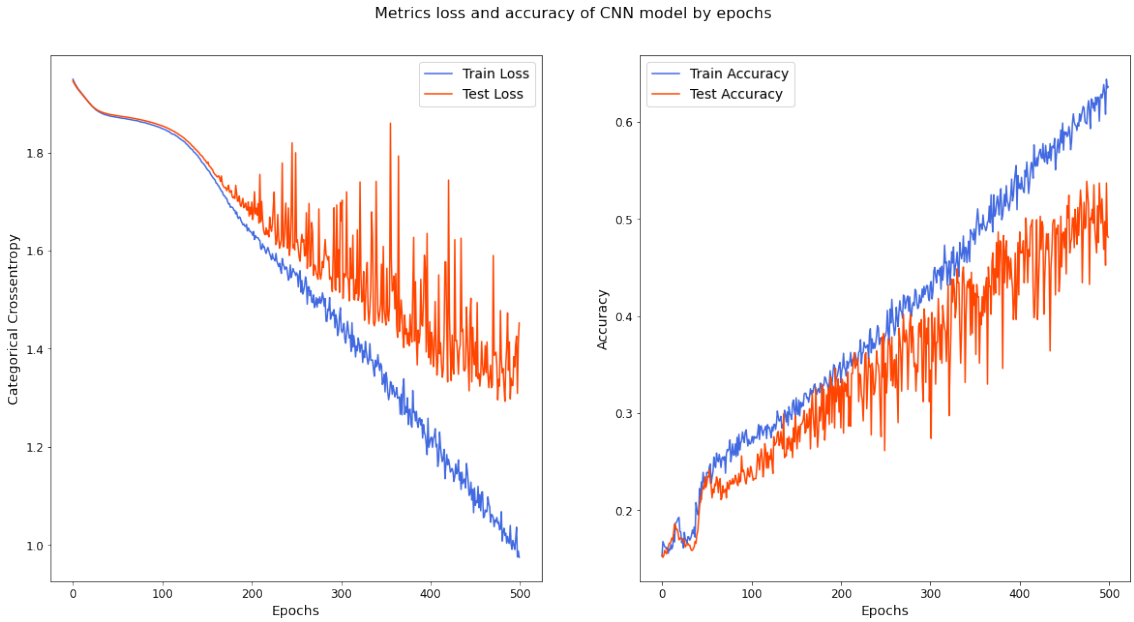

Figura 6.1: Resultados das métricas do modelo Fase de testes I - EXC 01

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.48108* e *Loss = 1.45250*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 348 (minutos)*.

6.1.1.2 Segunda execução

Em relação a segunda execução foi implementado a variação dos metaparâmetros *Optimizer = (SGD)* e *Learning_Rate = (0.0001)*, para realização do treinamento desse modelo. A Figura 6.2 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

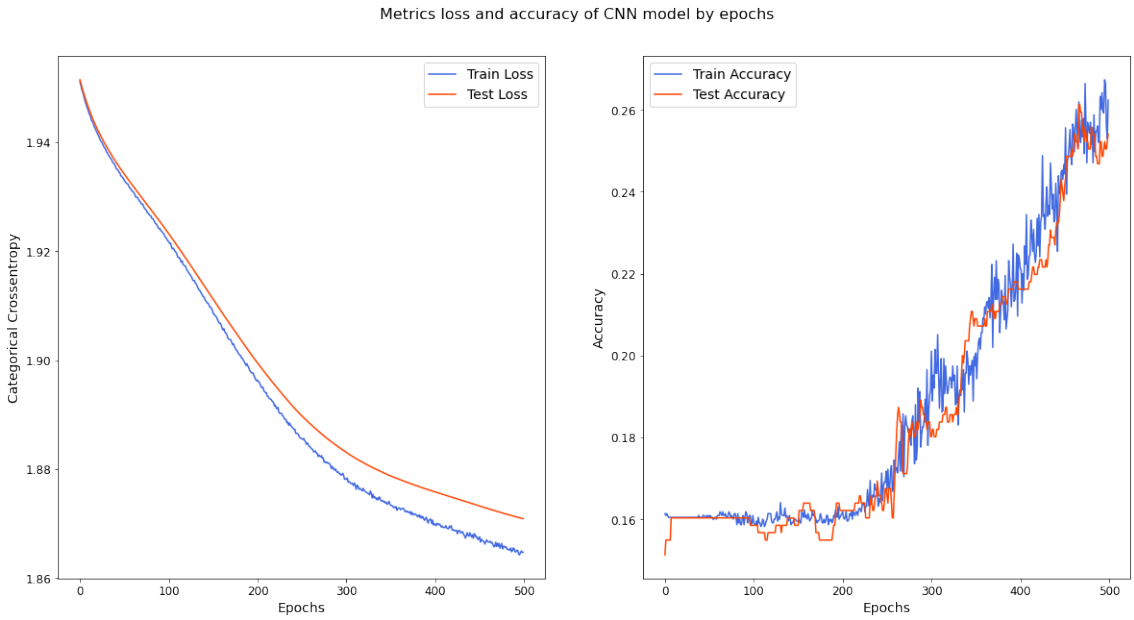

Figura 6.2: Resultados das métricas do modelo Fase de testes I - EXC 02

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.25405* e *Loss = 1.87090*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 353 (minutos)*.

6.1.1.3 Terceira execução

Em relação a terceira execução foi implementado a variação dos metaparâmetros *Optimizer = (SGD)* e *Learning_Rate = (0.00001)*, para realização do treinamento desse modelo. A Figura 6.3 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

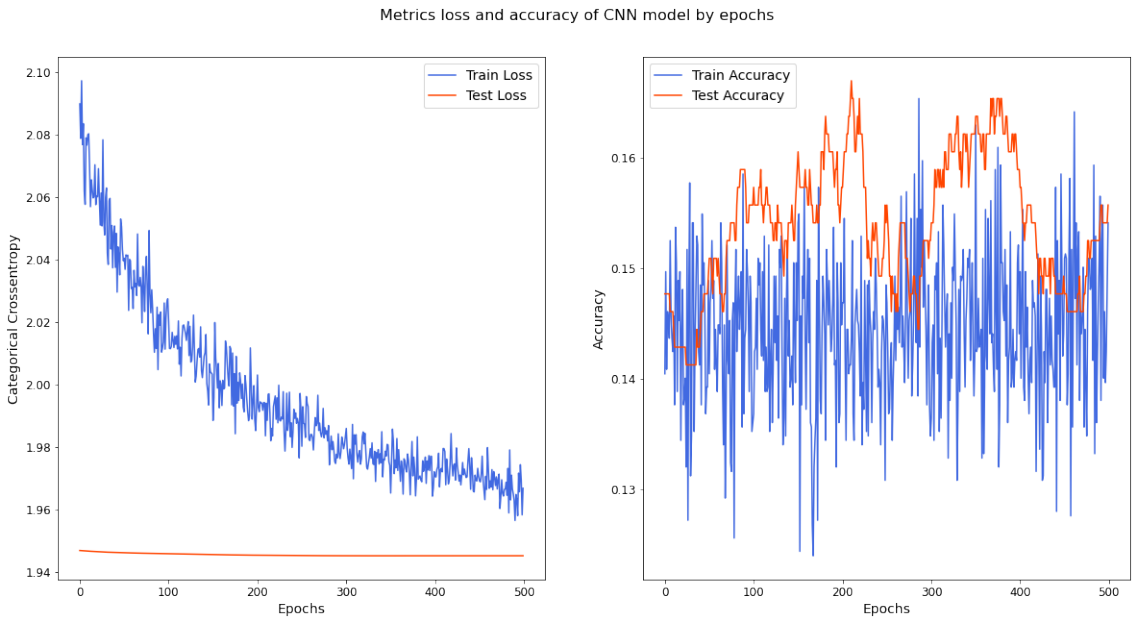

Figura 6.3: Resultados das métricas do modelo Fase de testes I - EXC 03

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.15569* e *Loss = 1.94511*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 344 (minutos)*.

6.1.1.4 Quarta execução

Em relação a quarta execução foi implementado a variação dos metaparâmetros *Optimizer = (ADAM)* e *Learning_Rate = (0.001)*, para realização do treinamento desse modelo. A Figura 6.4 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

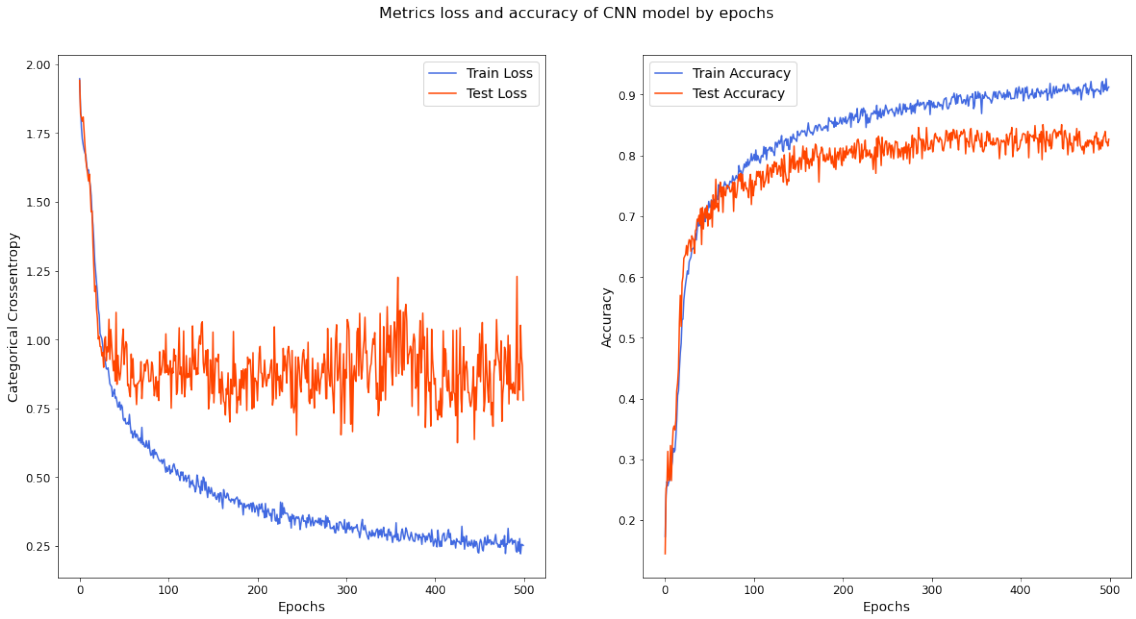

Figura 6.4: Resultados das métricas do modelo Fase de testes I - EXC 04

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.82664* e *Loss = 0.77918*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 358 (minutos)*.

6.1.1.5 Quinta execução

Em relação a quinta execução foi implementado a variação dos metaparâmetros *Optimizer = (ADAM)* e *Learning_Rate = (0.0001)*, para realização do treinamento desse modelo. A Figura 6.5 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

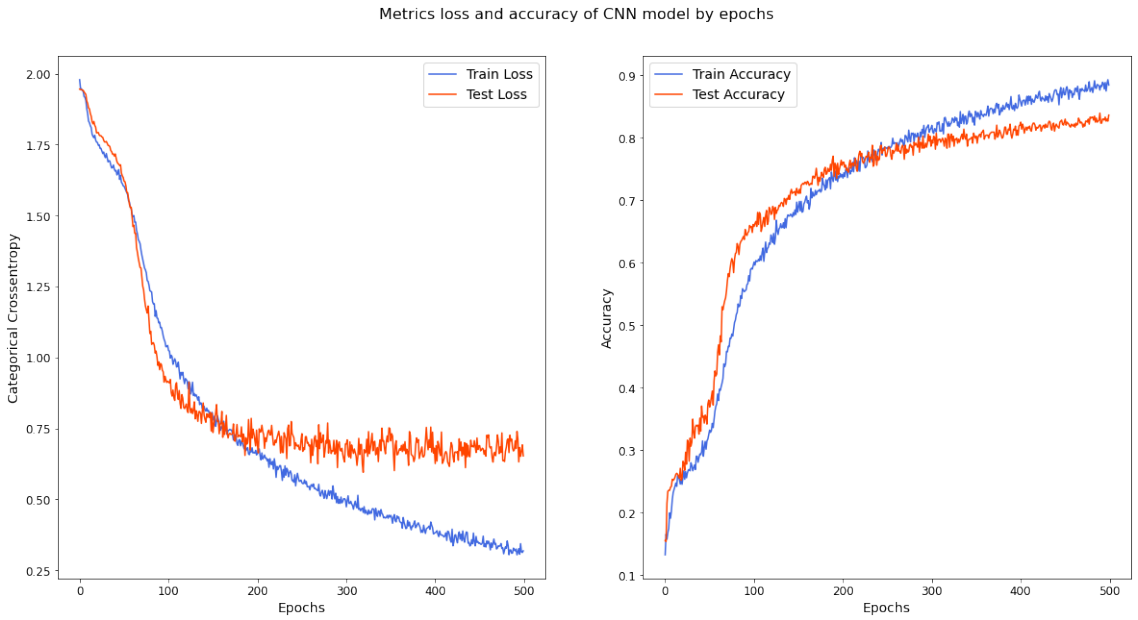

Figura 6.5: Resultados das métricas do modelo Fase de testes I - EXC 05

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.83627* e *Loss = 0.65294*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 307 (minutos)*.

6.1.1.6 Sexta execução

Em relação a sexta execução foi implementado a variação dos metaparâmetros *Optimizer = (ADAM)* e *Learning_ Rate = (0.00001)*, para realização do treinamento desse modelo. A Figura 6.6 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

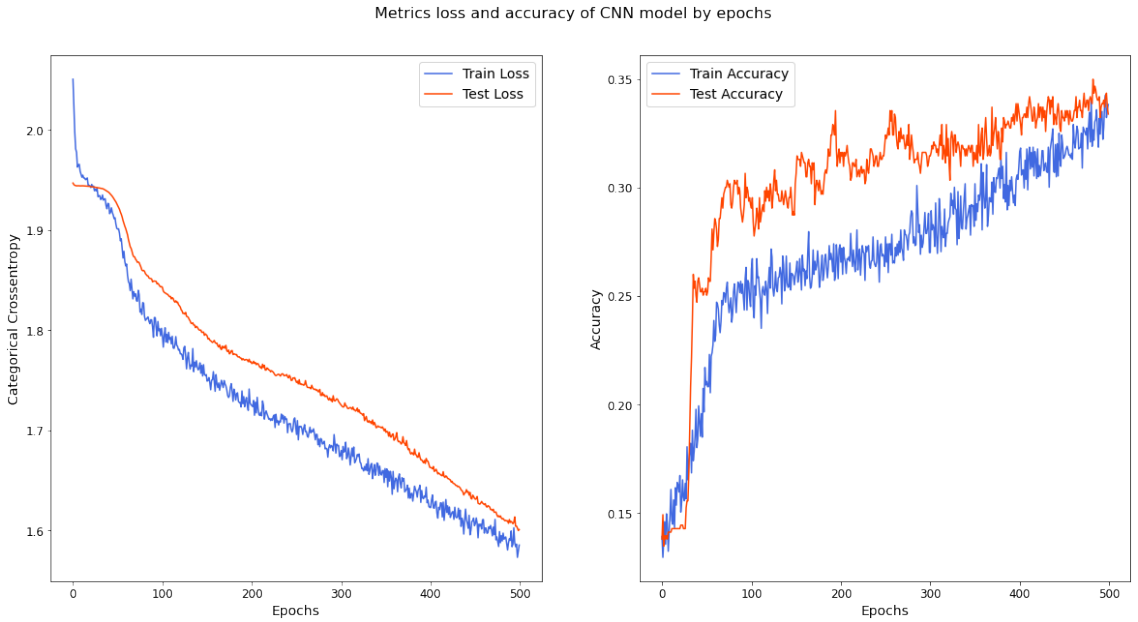

Figura 6.6: Resultados das métricas do modelo Fase de testes I - EXC 06

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.33386* e *Loss = 1.60094*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 329 (minutos)*.

6.1.1.7 Sétima execução

Em relação a sétima execução foi implementado a variação dos metaparâmetros *Optimizer = (RMSPROP)* e *Learning_Rate = (0.001)*, para realização do treinamento desse modelo. A Figura 6.7 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

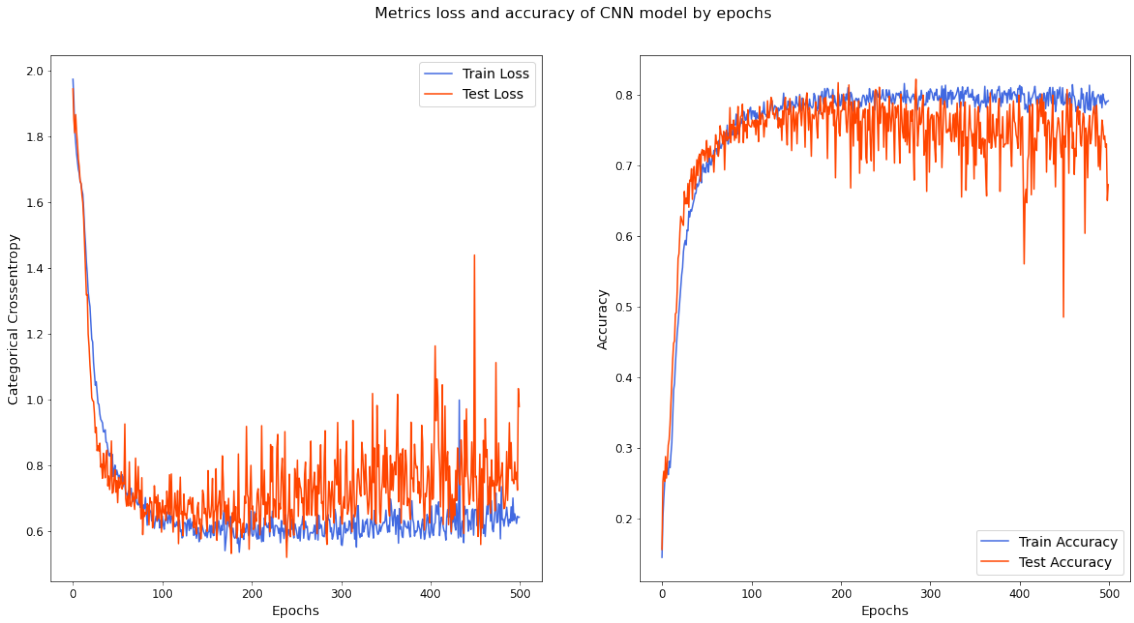

Figura 6.7: Resultados das métricas do modelo Fase de testes I - EXC 07

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.67255* e *Loss = 0.97788*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 345 (minutos)*.

6.1.1.8 Oitava execução

Em relação a oitava execução foi implementado a variação dos metaparâmetros *Optimizer = (RMSPROP)* e *Learning_Rate = (0.0001)*, para realização do treinamento desse modelo. A Figura 6.8 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

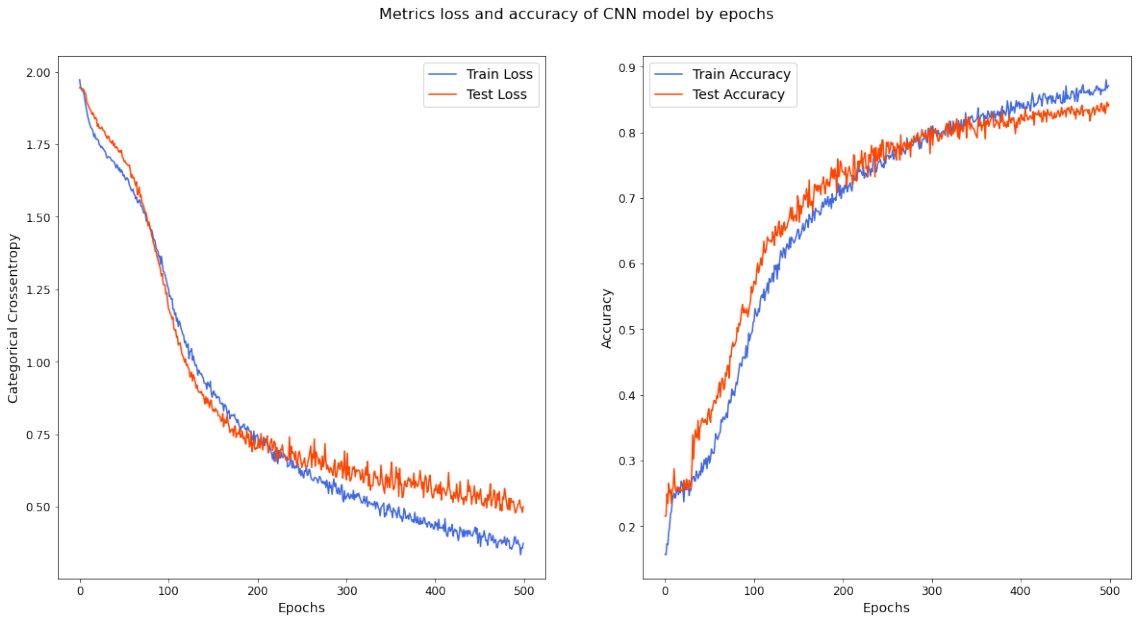

Figura 6.8: Resultados das métricas do modelo Fase de testes I - EXC 08

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.84269* e *Loss = 0.49949*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 356 (minutos)*.

6.1.1.9 Nona execução

Em relação a nona execução foi implementado a variação dos metaparâmetros *Optimizer = (RMSPROP)* e *Learning_Rate = (0.00001)*, para realização do treinamento desse modelo. A Figura 6.9 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

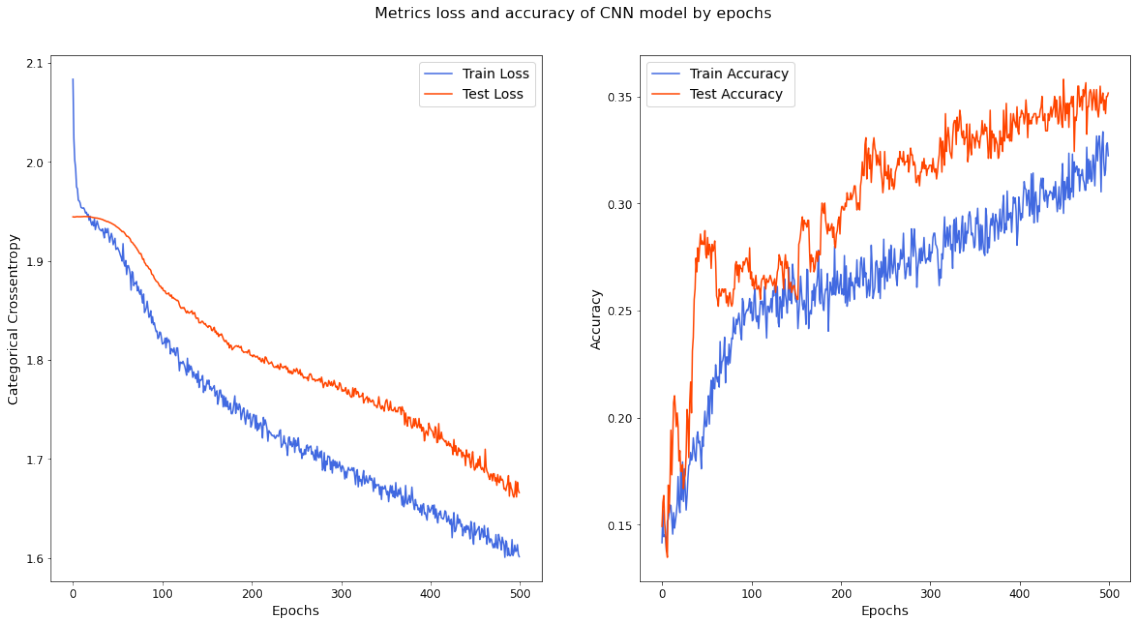

Figura 6.9: Resultados das métricas do modelo Fase de testes I - EXC 09

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.35152* e *Loss = 1.66614*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 353 (minutos)*.

6.1.2 Fase de testes II

Conforme descrito na Tabela 5.3, para realização da Fase de testes II serão realizadas seis execuções contemplando todas as variações de metaparâmetros que serão testados para o processo de finetunning. Ressaltando para essa fase de teste a configuração da arquitetura implementada: *CONFIG = CNN(3) - DNN(X,7)*, em que X é a variação de *Neurons = (10, 50, 100, 256, 512 e 1024)* referente a DNN.

6.1.2.1 Primeira execução

Em relação a primeira execução foi implementado a variação dos metaparâmetros *Neurons = (10)*, para realização do treinamento desse modelo. A Figura 6.10 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

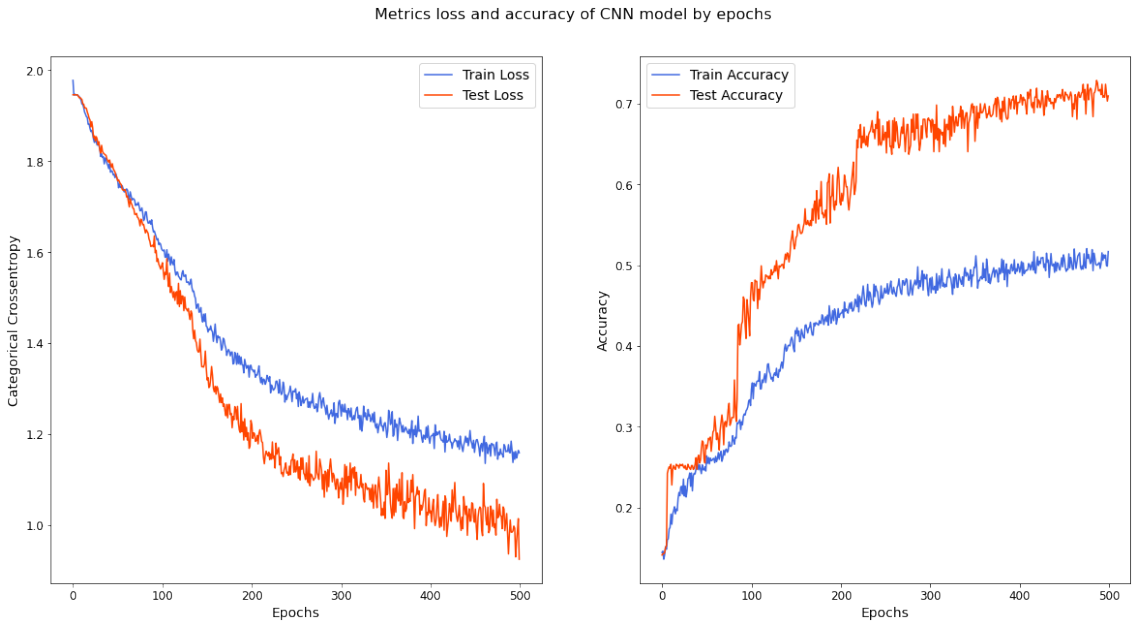

Figura 6.10: Resultados das métricas do modelo Fase de testes II - EXC 01

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.70947* e *Loss = 0.92487*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 186 (minutos)*.

6.1.2.2 Segunda execução

Em relação a segunda execução foi implementado a variação dos metaparâmetros *Neurons = (50)*, para realização do treinamento desse modelo. A Figura 6.11 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

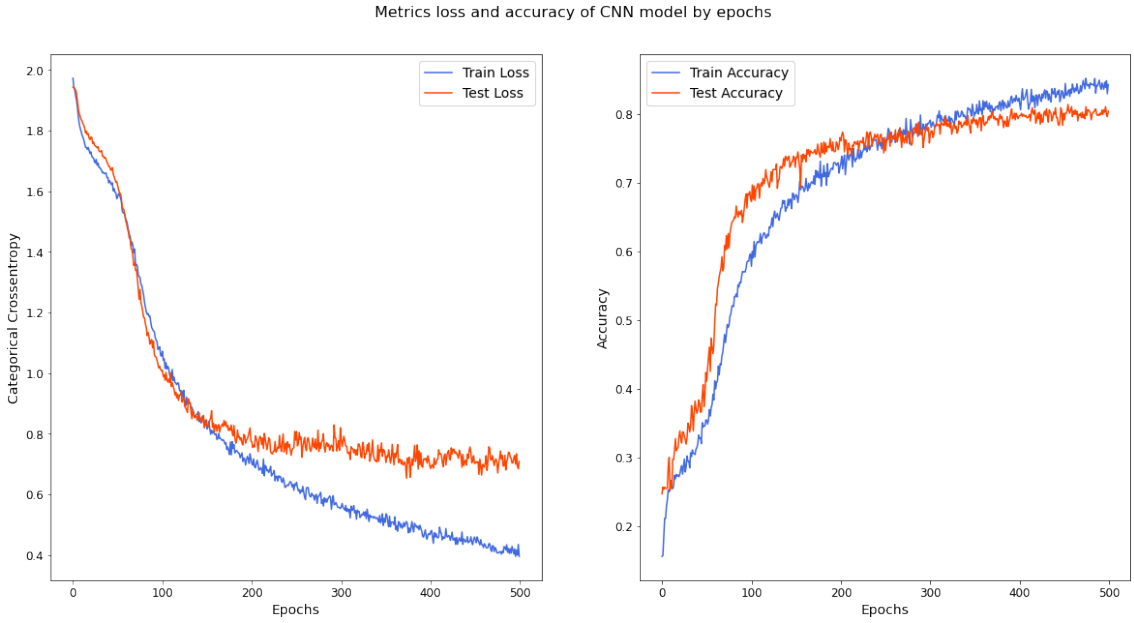

Figura 6.11: Resultados das métricas do modelo Fase de testes II - EXC 02

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.80417* e *Loss = 0.70806*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 190 (minutos)*.

6.1.2.3 Terceira execução

Em relação a terceira execução foi implementado a variação dos metaparâmetros *Neurons = (100)*, para realização do treinamento desse modelo. A Figura 6.12 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

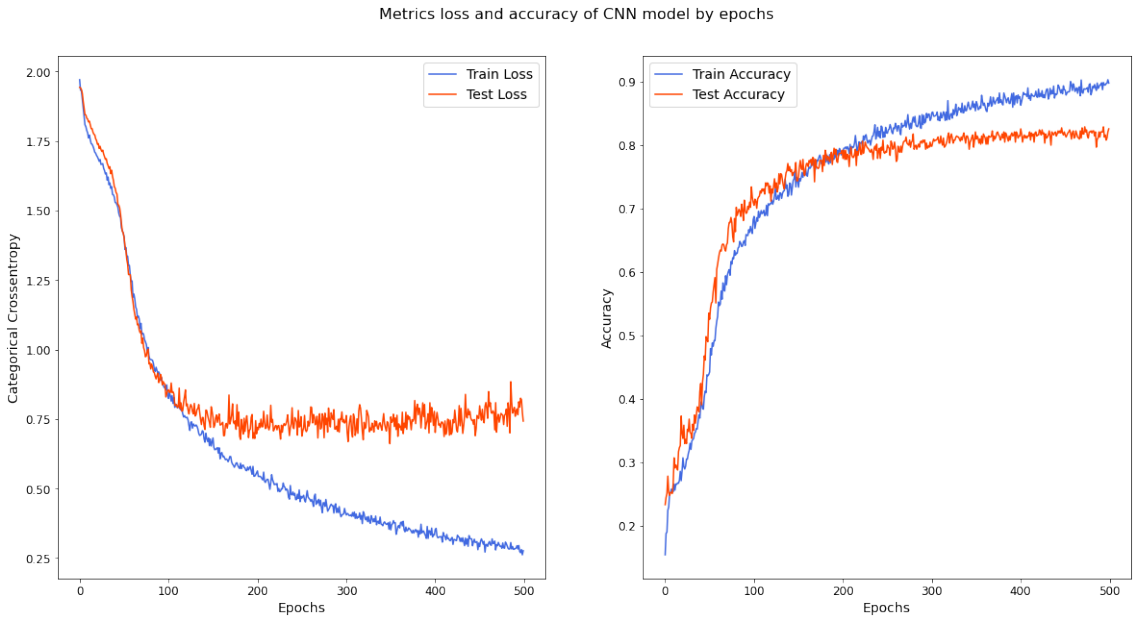

Figura 6.12: Resultados das métricas do modelo Fase de testes II - EXC 03

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.82504* e *Loss = 0.74253*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 183 (minutos)*.

6.1.2.4 Quarta execução

Em relação a quarta execução foi implementado a variação dos metaparâmetros *Neurons = (256)*, para realização do treinamento desse modelo. A Figura 6.13 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

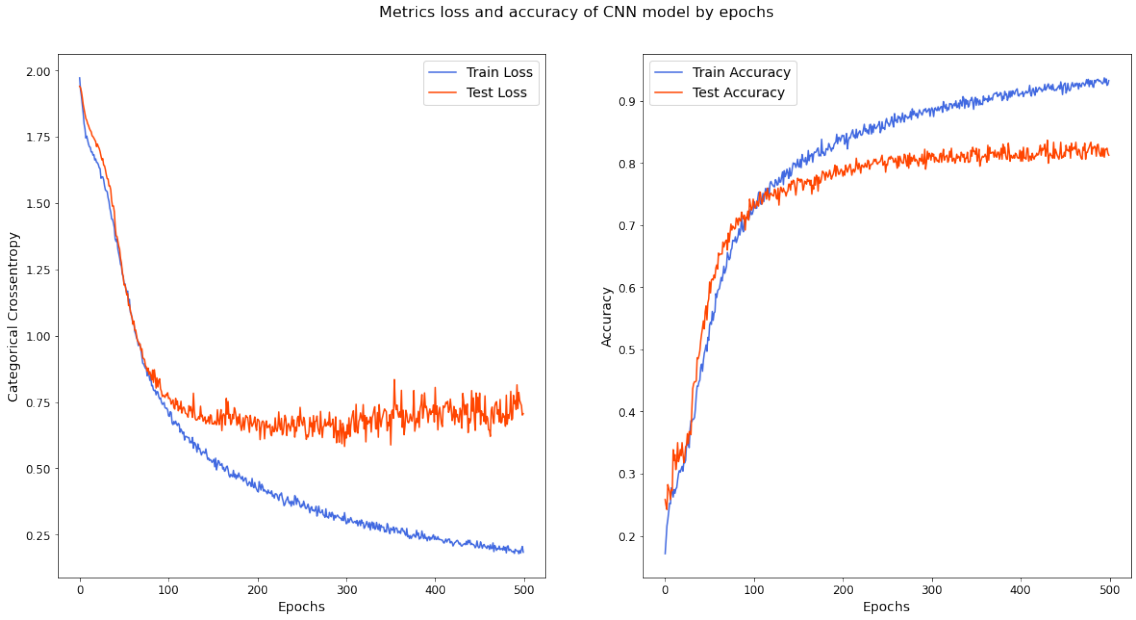

Figura 6.13: Resultados das métricas do modelo Fase de testes II - EXC 04

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.81219* e *Loss = 0.70566*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 181 (minutos)*.

6.1.2.5 Quinta execução

Em relação a quinta execução foi implementado a variação dos metaparâmetros *Neurons = (512)*, para realização do treinamento desse modelo. A Figura 6.14 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

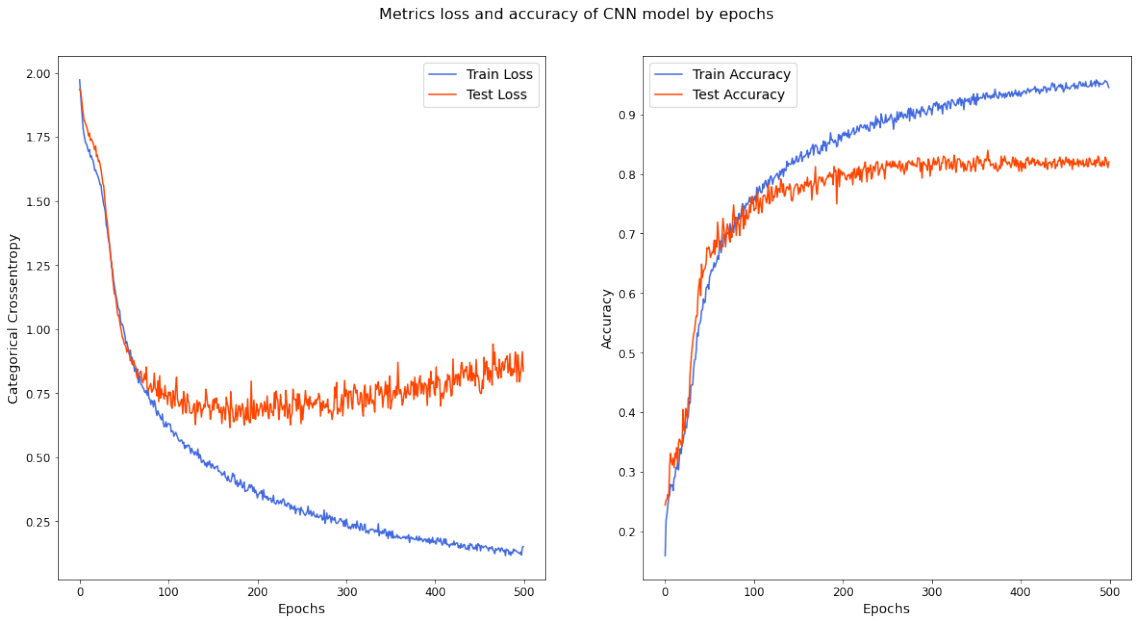

Figura 6.14: Resultados das métricas do modelo Fase de testes II - EXC 05

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.82022* e *Loss = 0.83552*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 188 (minutos)*.

6.1.2.6 Sexta execução

Em relação a sexta execução foi implementado a variação dos metaparâmetros *Neurons = (1024)*, para realização do treinamento desse modelo. A Figura 6.15 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

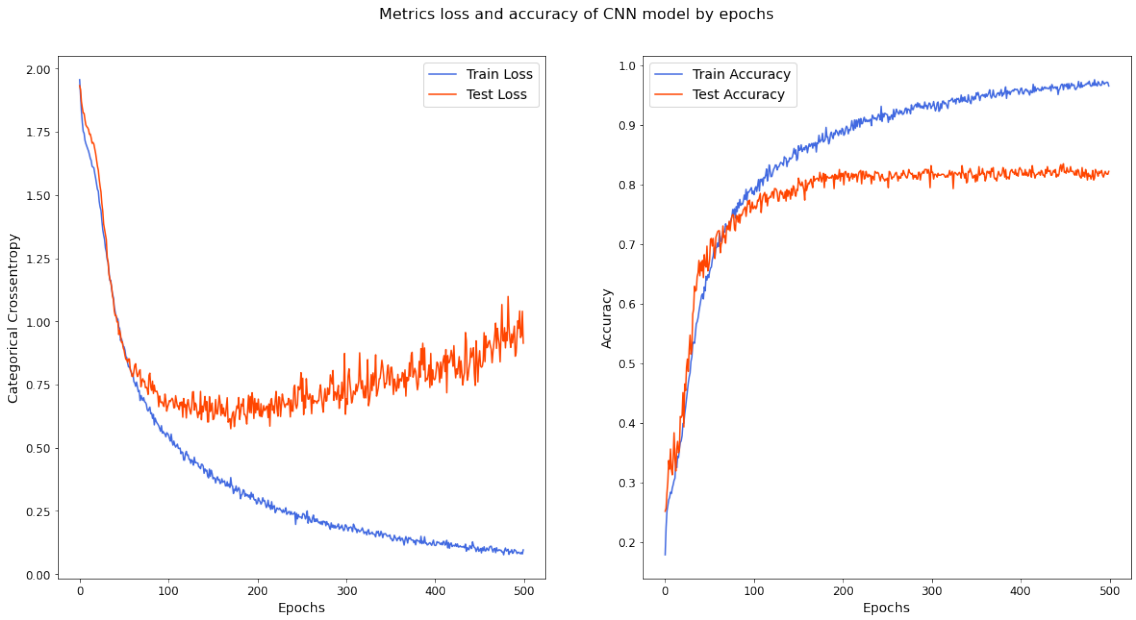

Figura 6.15: Resultados das métricas do modelo Fase de testes II - EXC 06

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.82182* e *Loss = 0.91362*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 195 (minutos)*.

6.1.3 Fase de testes III

Conforme descrito na Tabela 5.4, para realização da Fase de testes III serão realizadas quatro execuções contemplando todas as variações de metaparâmetros que serão testados para o processo de finetunning. Ressaltando para essa fase de teste a configuração da arquitetura implementada: *CONFIG = CNN(X) - DNN(256,256,7)*, em que X é a variação do número de convolutional layers *Convolutional layers = (2, 3, 4, 5)* referente a CNN.

6.1.3.1 Primeira execução

Em relação a primeira execução foi implementado a variação dos metaparâmetros *Convolutional layers = (2)*, para realização do treinamento desse modelo. A Figura 6.16 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

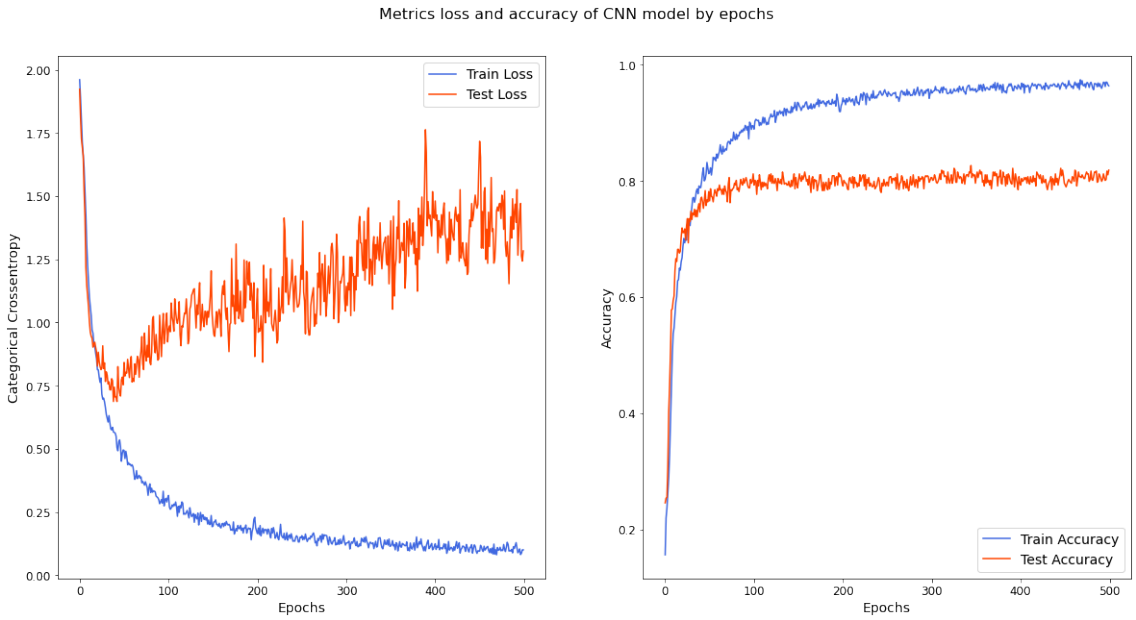

Figura 6.16: Resultados das métricas do modelo Fase de testes III - EXC 01

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.81861* e *Loss = 1.28328*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 264 (minutos)*.

6.1.3.2 Segunda execução

Em relação a segunda execução foi implementado a variação dos metaparâmetros *Convolutional layers = (3)*, para realização do treinamento desse modelo. A Figura 6.17 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

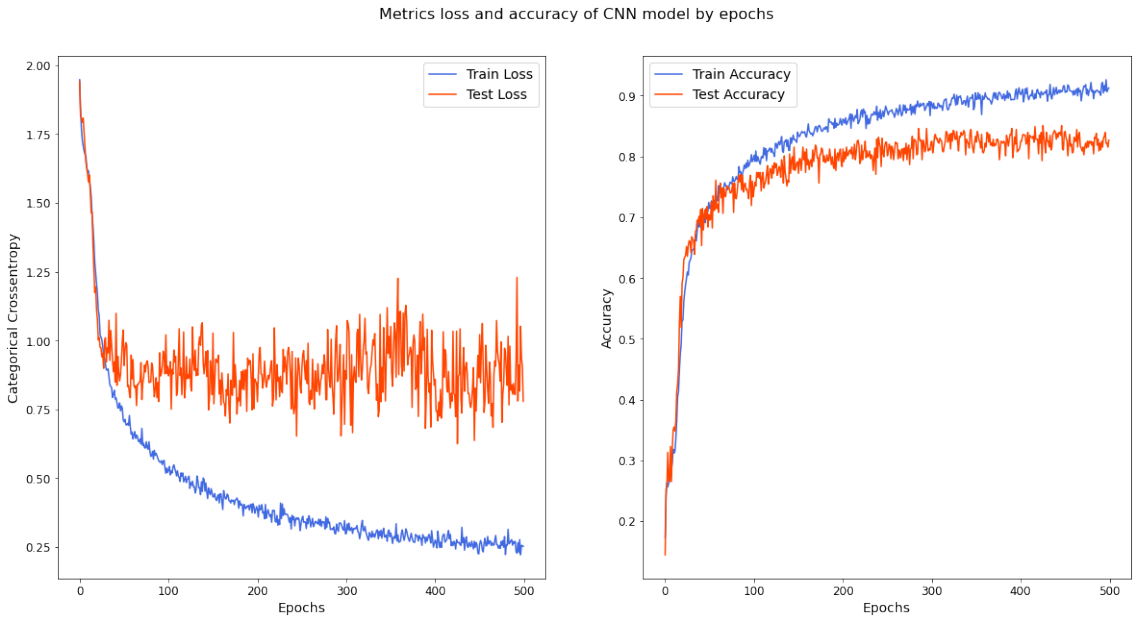

Figura 6.17: Resultados das métricas do modelo Fase de testes III - EXC 02

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.82664* e *Loss = 0.77918*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 377 (minutos)*.

6.1.3.3 Terceira execução

Em relação a terceira execução foi implementado a variação dos metaparâmetros *Convolutional layers = (4)*, para realização do treinamento desse modelo. A Figura 6.18 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

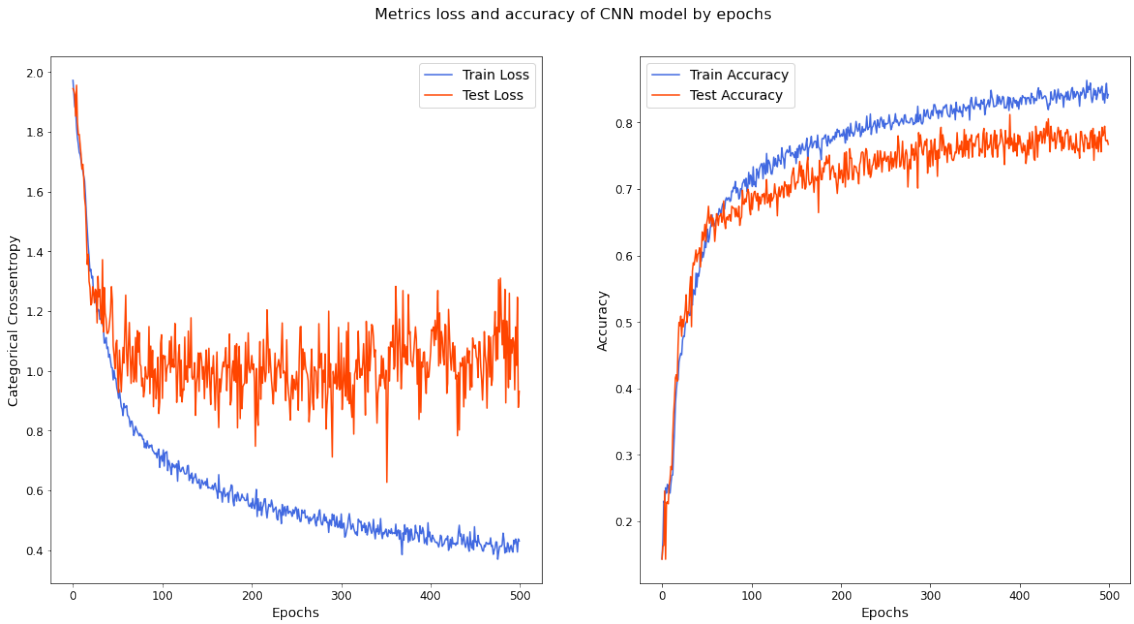

Figura 6.18: Resultados das métricas do modelo Fase de testes III - EXC 03

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.76725* e *Loss = 0.93161*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 483 (minutos)*.

6.1.3.4 Quarta execução

Em relação a quarta execução foi implementado a variação dos metaparâmetros *Convolutional layers = (5)*, para realização do treinamento desse modelo. A Figura 6.19 a seguir, apresenta o gráfico das métricas de performance (acurácia e loss) ao longo do treinamento desse modelo.

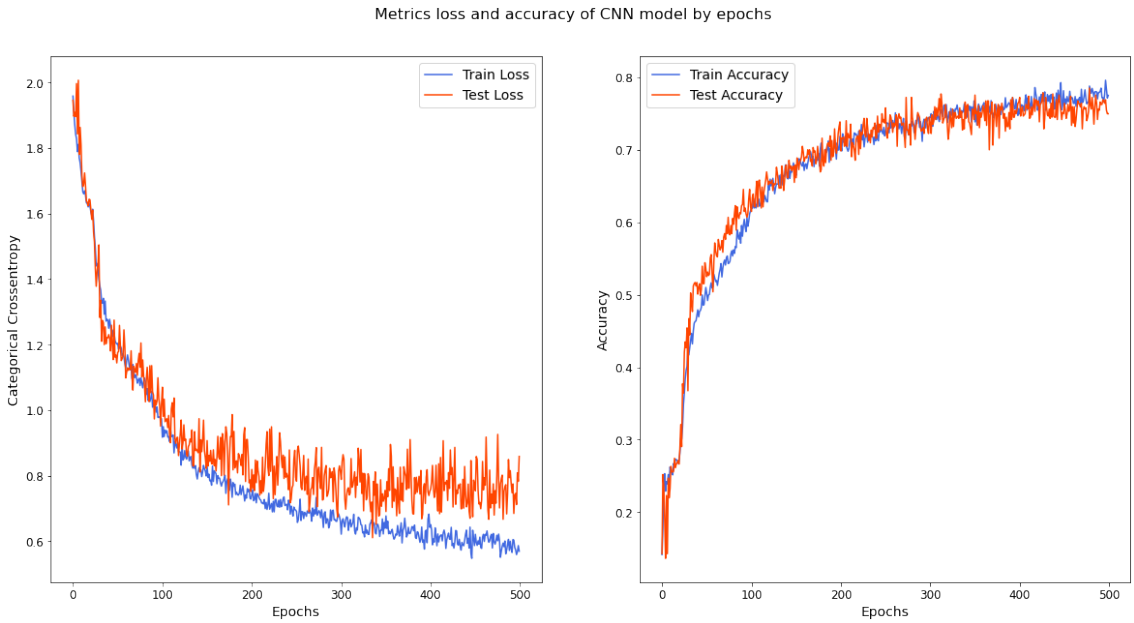

Figura 6.19: Resultados das métricas do modelo Fase de testes III - EXC 04

O resultado consolidado das métricas de performance para esse modelo foram de: *Acurácia = 0.77490* e *Loss = 0.56970*. Por fim, a métrica de custo computacional (CCP), ou seja, o tempo de treinamento desse modelo foi *CCP = 549 (minutos)*.

6.1.4 Análise dos resultados

Nesse momento serão avaliados os resultados obtidos dos modelos de Classificadores Individuais, além de destacar os melhores a partir da performance das métricas de avaliação. O objetivo ao fim dessa análise é averiguar o segundo objetivo específico dessa dissertação: "Determinar a melhor arquitetura e metaparâmetros para a tarefa de classificação de emoções humanas básicas a partir de imagens digitais, realizando o processo de finetunning da rede proposta utilizando algoritmos de CNNs".

6.1.4.1 Fase de testes I

Inicialmente durante a Fase de testes I, o objetivo foi de avaliar os metaparamêtros de Optimizer function e Leaning Rate durante nove execuções de treinamento com todas as variações estabelecidas. A Tabela 6.1 a seguir, consolida os resultados das métricas de avaliação (Acurácia), (Loss) e Custo computacional (CCP) em minutos obtidas para essa fase de testes em relação a cada execução.

Seguindo a referência da Tabela 6.1, pode ser observado para a variação utilizando o algoritmo SGD, execuções 1-3 o melhor resultado obtido foi com a combinação de *LR = (0.001)* resultando numa acurácia final igual a 48,108%. Já para as outras learning rates observou-se uma queda do desempenho do modelo. Comprovando-se que a diminuição da learning rate para o caso do algoritmo SGD compromete a acurácia do resultado.

Tabela 6.1: Resultados das métricas de avaliação da Fase de testes I

Execução	**Acurácia**	**Loss**	**CCP (min)**
01	0.48108	1.45250	348
02	0.25405	1.87090	353
03	0.15569	1.94511	344
04	0.82664	0.77918	358
05	0.83627	0.65294	305
06	0.33386	1.60094	329
07	0.67255	0.97788	345
08	0.84269	0.49949	356
09	0.35152	1.66614	353

Já para a variação com algoritmo ADAM, execuções 4-6 o melhor resultado atingido foi com a combinação de *LR = (0.0001)* resultando numa acurácia final igual a 83,627% também com a menor métrica de loss na comparação das três execuções. Destaca-se também o resultado com a combinação de *LR = (0.001)* que também obteve um resultado bem próximo. E uma queda no desempenho ocorrido com a utilização da LR = (0.00001), observa-se um desempenho em geral melhor com utilização do algoritmo ADAM em comparação ao SGD.

Por último para a variação com algoritmo RMSPROP, execuções 7-9 o melhor resultado atingido foi com a combinação de *LR = (0.0001)* resultando numa acurácia final igual a 84,269% também com a menor métrica de loss na comparação das três execuções. Conclui-se para o caso do algoritmo RMSPROP, houve uma obtenção de resultados próximos das execuções com ADAM.

Por tanto, com os dados das variações dos algoritmos (SGD, ADAM, RMSPROP) classifica-se como o melhor desempenho obtido na execução cinco com utilização do algoritmo ADAM em combinação com uma *LR = (0.0001)*, obtendo uma acurácia final igual a 83,627%. Essa classificação também leva em conta a segunda métrica de loss mais baixa obtida entre todas execuções, além da métrica de custo computacional mais baixa observada com um tempo de treinamento total igual a 305 minutos.

Entretanto, vale-se a menção a execução oito utilizando algoritmo RMSPROP obtendo excelentes métricas de performance, mas com um dos custos computacionais mais altos. Conclui-se da análise que os casos utilizando ADAM e RMSPROP obtiveram resultados similares, entretanto o SGD destoa caindo a performance do modelo com diminuição da acurácia, principalmente para learning rates menores.

6.1.4.2 Fase de testes II

Já para Fase de testes II, o objetivo foi de avaliar o metaparamêtro de Neurons em relação a rede DNN durante seis execuções de treinamento com todas as variações estabelecidas. A Tabela 6.2 a seguir, consolida os resultados das métricas de avaliação (Acurácia), (Loss) e Custo computacional (CCP) em minutos obtidas para essa fase de testes em relação a cada execução.

Tabela 6.2: Resultados das métricas de avaliação da Fase de testes II

Execução	**Acurácia**	**Loss**	**CCP (min)**
01	0.70947	0.92487	186
02	0.80417	0.70806	190
03	0.82504	0.74253	183
04	0.81219	0.70566	181
05	0.82022	0.83552	188
06	0.82182	0.91362	195

Conclui-se das execuções que os resultados obtidos ficaram muito próximos em relação a média de acurácia, excluisvamente a primeira execução, variação com *(Neurons = 10)* destoa com uma queda na performance do modelo. Cassifica-se como o melhor desempenho obtido na execução quatro com utilização da variação com *(Neurons = 256)*, atingindo uma acurácia final igual 81,219%, justificada também por atingir a métrica de loss mais baixa entre todas execuções e métrica de custo computacional mais baixa observada com um tempo de treinamento total igual a 181 minutos.

6.1.4.3 Fase de testes III

Já para Fase de testes III, o objetivo foi de avaliar os metaparamêtros de Convolutional Layers em relação a rede CNN durante quatro execuções de treinamento com todas as variações estabelecidas. A Tabela 6.3 a seguir, consolida os resultados das métricas de avaliação (Acurácia), (Loss) e Custo computacional (CCP) em minutos obtidas para essa fase de testes em relação a cada execução.

Tabela 6.3: Resultados das métricas de avaliação da Fase de testes III

Execução	**Acurácia**	**Loss**	**CCP (min)**
01	0.81861	1.28328	264
02	0.82664	0.77918	377
03	0.76725	0.93161	483
04	0.77490	0.56970	549

Com os dados obtidos dos resultados das execuções para essa última fase de testes, é necessário analisar todas as métricas de performance com objetivo de avaliar o melhor desempenho. Concluise que para o melhor resultado observado, ainda que métricas de acurácia mais altas foram obtidas para a variação com três camadas de Convolutional layers, a execução quatro com a variação de cinco camadas de Convolutional layers detém o melhor resultado em relação a métrica de loss, ou seja, o modelo com menor erro durante a validação. Atentando-se ao gráfico dessa execução que possui o melhor comportamento ao longo das 500 épocas de treinamento e atingindo o menor valor também comparado ao restante das execuções.

6.1.5 Seleção dos Classificadores Individuais

Nesse momento já com os resultados analisados das execuções dos modelos de Classificadores Individuais, é possível realizar o destaque das melhores performances obtidas e seleção dos modelos para composição do Módulo Pool de Classificadores.

Dessa forma, serão selecionados cinco modelos de Classificadores Individuais a partir das execuções já apresentadas em um ranking considerando-se a melhores performances. A Tabela 6.4 a seguir, detalhas os modelos de Classificadores Individuais selecionados para composição do Pool de classificadores da solução SISCOM.

Tabela 6.4: Seleção dos modelos de Classificadores Individuais

	Arquitetura	**Optimizer**	**Learning Rate**	**DNN Neurons**
CL_01	CNN(3) - DNN(128,128,7)	SGD	0.001	128
CL_02	CNN(3) - DNN(128,128,7)	ADAM	0.0001	128
CL_03	CNN(3) - DNN(128,128,7)	RMSPROP	0.0001	128
CL_04	CNN(3) - DNN(256,7)	ADAM	0.0001	256
CL_05	CNN(5) - DNN(256,256,7)	ADAM	0.001	256

6.2 Sistema Comitê SISCOM

Nessa subseção serão detalhados os resultados obtidos para as duas configurações implementadas da solução SISCOM. Dessa forma, na sequência dessa dissertação foi organizado a apresentação desses resultados conforme (SISCOM I e SISCOM II), também serão analisadas as métricas de performance obtidas. A expectativa ao fim dessa etapa é esclarecer os objetivos específicos dessa dissertação: *"Desenvolver um Sistema Comitê com os Classificadores Individuais gerados com o objetivo de aumentar as métricas de performance da rede"* e *"Determinar o melhor número de Classificadores Individuais integrados para composição do Sistema Comitê"*.

6.2.1 SISCOM I

Conforme apresentado na análise dos resultados dos modelos de Classificadores Individuais, a Tabela 6.4 detalha os modelos selecionados para composição do Módulo pool de classificadores da solução SISCOM. Para o caso da configuração SISCOM I, serão utilizados três modelos de Classificadores Individuais para composição do seu pool de classificadores.

Visando garantir que exista diversidade no conjunto do pool de classificadores, a estratégia utilizada foi de selecionar um modelo de cada fase de testes garantindo diferentes parâmetros de treinamento, considerando um mesmo classificador, porém com diferentes configurações.

Por tanto, seguindo a referência da Tabela foram utilizados para composição do SISCOM I os modelos de Classificadores Individuais *(CL_02, CL_04, CL_05)*. O trecho de código 6.1 a seguir

detalha essa consolidação do SISCOM I, destaca-se a linha 16 em que ocorre a atribuição dos pesos referentes a cada modelo, considerando a performance obtida.

```
1  from keras.models import load_model
2  from sklearn.metrics import accuracy_score
3  import numpy as np
4
5  # loading models
6  model1 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft01_exc05.h5')
7  model2 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft02_exc04.h5')
8  model3 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft03_exc04.h5')
9
10 # combine predictions
11 predictions1 = model1.predict(test_set)
12 predictions2 = model2.predict(test_set)
13 predictions3 = model3.predict(test_set)
14
15 # ensemble method weight average
16 weights = [0.5, 0.2, 0.3]
17 preds = np.array([predictions1, predictions2, predictions3])
18 weighted_preds = np.tensordot(preds, weights, axes=((0),(0)))
19 weighted_ensemble_prediction = np.argmax(weighted_preds, axis=1)
20
21 # calculate ensemble accuracy
22 ensemble_accuracy = accuracy_score(test_set.labels, weighted_ensemble_prediction)
23 print('Accuracy Score for average weight ensemble model = ', ensemble_accuracy)
```

Trecho de código 6.1: Consolidação do Módulo pool de classificadores SISCOM I

Com essa consolidação da solução SISCOM I, o resultado obtido para essa configuração com três Classificadores Individuais compondo o pool de classificadores foi de uma métrica de acurácia igual a 95,248% e um custo computacional de 27 minutos.

Esse excelente resultado de custo computacional era esperado pela construção em módulos da solução SISCOM, de forma que não ocorre treinamento dos modelos nessa etapa. Ocorre apenas o carregamento dos modelos já treinados e uma operação matemática que combina as predições de cada modelo de classificador aos pesos atribuídos numa classificação final. Conclui-se a da métrica de acurácia que a técnica de Sistemas Múltiplos Classificadores é uma excelente estratégia visando a melhora da performance e generalização combinando diferentes modelos para uma classificação final.

6.2.2 SISCOM II

Conforme apresentado na análise dos resultados dos modelos de Classificadores Individuais, a Tabela 6.4 detalha os modelos selecionados para composição do Módulo pool de classificadores da solução SISCOM. Para o caso da configuração SISCOM II, serão utilizados cinco modelos de

Classificadores Individuais para composição do seu pool de classificadores.

Por tanto, seguindo a referência da Tabela foram utilizados para composição do SISCOM II todos os modelos de Classificadores Individuais *(CL_01, CL_02, CL_03, CL_04, CL_05)*. O trecho de código 6.2 a seguir detalha essa consolidação do SISCOM II, destaca-se a linha 16 em que ocorre a atribuição dos pesos referentes a cada modelo, considerando a performance obtida.

```
1  from keras.models import load_model
2  from sklearn.metrics import accuracy_score
3  import numpy as np
4
5  # loading models
6  model1 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft01_exc01.h5')
7  model2 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft01_exc05.h5')
8  model3 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft01_exc08.h5')
9  model4 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft02_exc04.h5')
10 model5 = load_model('/content/drive/MyDrive/Colab Notebooks/PFG2/saved_models/
       modelo_ft03_exc04.h5')
11
12 # combine predictions
13 predictions1 = model1.predict(test_set)
14 predictions2 = model2.predict(test_set)
15 predictions3 = model3.predict(test_set)
16 predictions4 = model4.predict(test_set)
17 predictions5 = model5.predict(test_set)
18
19 # ensemble method weight average
20 weights = [0.3, 0.2, 0.2, 0.2, 0.1]
21 preds = np.array([predictions1, predictions2, predictions3, predictions4,
       predictions5])
22 weighted_preds = np.tensordot(preds, weights, axes=((0),(0)))
23 weighted_ensemble_prediction = np.argmax(weighted_preds, axis=1)
24
25 # calculate ensemble accuracy
26 ensemble_accuracy = accuracy_score(test_set.labels, weighted_ensemble_prediction)
27 print('Accuracy Score for average weight ensemble model = ', ensemble_accuracy)
```

Trecho de código 6.2: Consolidação do Módulo pool de classificadores SISCOM II

Com essa consolidação da solução SISCOM II, o resultado obtido para essa configuração com cinco Classificadores Individuais compondo o pool de classificadores foi de uma métrica de acurácia igual a 95,569% e um custo computacional de 49 minutos.

Realizando a comparação com os resultados obtidos com a solução SISCOM I, observa-se métricas de performance extremamente próximas. Ressalta-se o detalhe para essa configuração com cinco modelos de Classificadores Individuais, houve um aumento do custo computacional aproximadamente em 1,814%, já a métrica de acurácia quase manteve-se a mesma.

6.3 Análises Estatísticas

Nesta seção, os resultados das questões de pesquisa são abordados, exibindo os resultados obtidos por sete classificadores. Dentre eles, cinco são classificadores individuais, enquanto os outros dois são comitês formados por CNNs com diferentes metaparâmetros.

Os classificadores que apresentarem a maior acurácia e menor tempo de execução em relação a cada base de dados serão comparados com os demais por meio de testes estatísticos. Essa análise permitirá identificar o modelo mais adequado para cada situação, assim como seus pontos positivos e negativos.

No contexto desse estudo, o teste de normalidade de *Shapiro-Wilk* foi empregado para avaliar estatisticamente a natureza da distribuição dos dados, utilizando um nível de confiança de 95% (α=0.05). Para verificar se há diferença significativa entre as amostras, serão empregados os testes de *t-Student* e de *Wilcoxon*, de acordo com a distribuição dos dados. Foram realizados dez execuções de cada classificador para coletar uma amostra possível de avaliar nesse estudo.

Na Tabela 6.5, encontram-se os detalhes sobre a acurácia alcançada por cada um dos classificadores em relação à base de dados. Adicionalmente, são apresentadas informações sobre o tempo de execução em minutos de cada classificador, indicando o intervalo de tempo que levou para realizar as classificações.

Tabela 6.5: Performance dos Classificadores e Comitês

Classificador	**Acurácia (%)**	**Tempo (min)**
CL_01	48,108	348
CL_02	83,627	305
CL_03	84,269	356
CL_04	81,219	181
CL_05	77,490	549
SISCOM I	95,248	**27**
SISCOM II	**95,569**	49

Antes de escolher o teste de hipótese adequado para comparação entre amostras, é necessário verificar a natureza da distribuição dos dados. Especialmente quando se trata de uma amostra pequena, com um tamanho de n = 10, a realização de um teste de normalidade, como o teste de Shapiro-Wilk, é fundamental. Os resultados deste teste de normalidade são fornecidos na Tabela 6.6, permitindo avaliar se os dados seguem ou não uma distribuição normal. Com base nessa análise, será possível selecionar o procedimento estatístico mais apropriado para realizar as comparações entre as amostras.

Dentre todos os classificadores avaliados, nenhum é possível rejeitar a hipótese nula. Isso significa que o teste de hipótese utilizado é o teste de t-student. Com o conhecimento da distribuição dos dados através do teste de normalidade, o próximo passo é verificar se existe diferença estatística significativa entre os desempenhos dos classificadores. Uma vez que o SISCOM II obteve a maior

Tabela 6.6: Teste de Normalidade dos Classificadores

Classificador	**p-value**	**p-value** $< \alpha$	**Resultado**
CL_01	0.92965	Não	Não é possível rejeitar H_0
CL_02	0.09473	Não	Não é possível rejeitar H_0
CL_03	0.86490	Não	Não é possível rejeitar H_0
CL_04	0.43080	Não	Não é possível rejeitar H_0
CL_05	0.2999	Não	Não é possível rejeitar H_0
SISCOM I	0.54705	Não	Não é possível rejeitar H_0
SISCOM II	0.50613	Não	Não é possível rejeitar H_0

acurácia, seu desempenho será comparado com os desempenhos dos outros modelos. O SISCOM I também será comparado aos outros por ter o menor tempo de execução.

Como todas as amostras são de uma distribuição normal, o teste de hipótese usado será o t-student, tendo como $\alpha = 0.05$ e como hipóteses:

- H_0: As médias das duas amostras são iguais
- H_1: As médias das duas amostras são diferentes.

Tabela 6.7: Comparação da Performance entre SISCOM II e demais Classificadores

Classificador	**p-value**	**p-value** $< \alpha$	**Resultado**
SISCOM II x CL_01	6.9355e-22	Sim	Rejeita H_0
SISCOM II x CL_02	8.6169e-15	Sim	Rejeita H_0
SISCOM II x CL_03	1.8761e-15	Sim	Rejeita H_0
SISCOM II x CL_04	4.8802e-16	Sim	Rejeita H_0
SISCOM II x CL_05	8.5627e-15	Sim	Rejeita H_0
SISCOM II x SISCOM I	0.3734	Não	Não é possível rejeitar H_0

Tabela 6.8: Comparação da Performance entre SISCOM I e demais Classificadores

Classificador	**p-value**	**p-value** $< \alpha$	**Resultado**
SISCOM I x CL_01	3.0622e-22	Sim	Rejeita H_0
SISCOM I x CL_02	4.2896e-15	Sim	Rejeita H_0
SISCOM I x CL_03	9.0567e-16	Sim	Rejeita H_0
SISCOM I x CL_04	7.4901e-17	Sim	Rejeita H_0
SISCOM I x CL_05	5.7276e-15	Sim	Rejeita H_0
SISCOM I x SISCOM II	0.3734	Não	Não é possível rejeitar H_0

Por meio das Tabelas 6.7 e 6.8, é possível analisar que os SISCOMs demonstram um desempenho estatisticamente superior em relação a todos os outros modelos classificadores. Para uma análise

mais aprofundada das diferenças nas acurácias dos classificadores, a Figura 6.20 mostra por meio de box-plots, a comparação da distribuição das acurácias entre os SISCOMs e os demais classificadores. Esses gráficos permitem visualizar a dispersão e a mediana das acurácias, bem como possíveis valores discrepantes, o que ajuda a entender melhor a variação nos resultados entre os modelos.

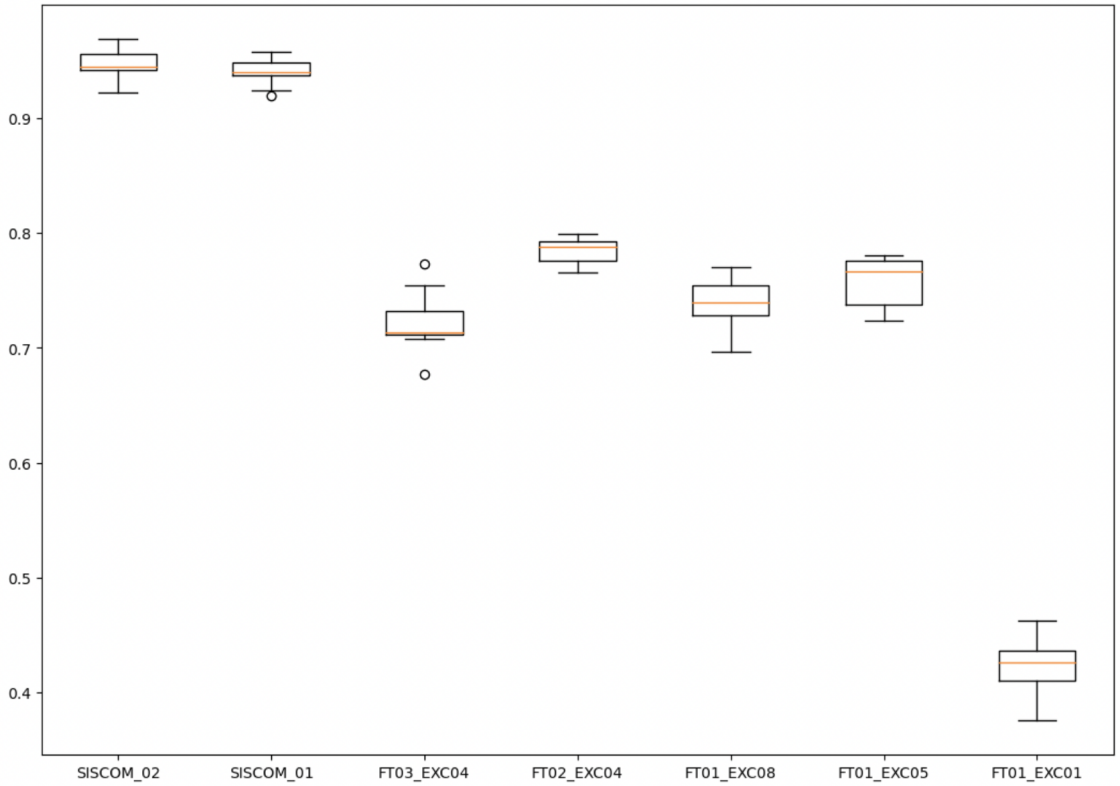

Figura 6.20: Comparação da distribuição das acurácias entre os Classificadores

Nota-se pela Figura 6.20 que além da acurácia, outras métricas relevantes ao analisar um gráfico boxplot são melhores nos SISCOMs comparados aos classificadores individuais. O intervalo interquartílico é menor nos SISCOMs, indicando uma menor variabilidade (dispersão) do modelo, as caudas tem uma diferença menor entre elas e apenas no SISCOM_01 encontra-se um valor discrepante.

Essas informações são fundamentais para destacar a superioridade estatística dos SISCOMs em relação aos outros classificadores e para fornecer uma visão mais clara sobre a distribuição das acurácias, possibilitando uma avaliação abrangente do desempenho de cada modelo. Na figura 6.21 mostra a distribuição das acurácias entre os SISCOMs, ficando clara a semelhança entre os dois modelos.

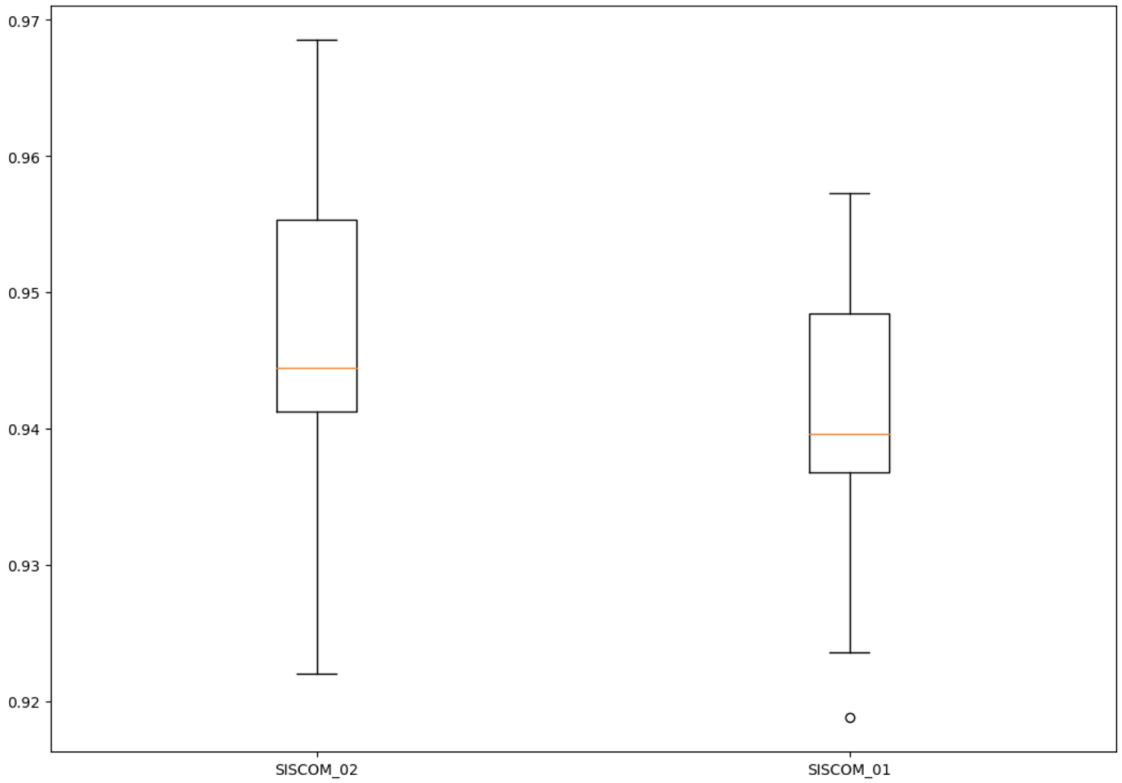

Figura 6.21: Comparação da distribuição das acurácias entre os SISCOMs

Pela Figura 6.21, pode-se observar a semelhança entre os dois SISCOMs porém nota-se suas leves diferenças. O SISCOM_01 tem uma dispersão menor dos seus dados e uma diferença entre as caudas menor, apesar de ter em sua distribuição um valor discrepante. Os dois comitês tem seus dados assimétricos positivos, pela mediana estar mais próxima da parte inferior da caixa (primeiro quartil).

Neste segmento, foram apresentados os resultados dos classificadores treinados nesta dissertação, e uma análise estatística foi conduzida para comparar o desempenho de cada um. Os SISCOMs demonstraram um desempenho superior em relação aos demais classificadores, sendo que o SISCOM II obteve a maior acurácia e o SISCOM I apresentou o menor tempo de execução. Considerando ambos os valores, pode-se afirmar que o SISCOM I é uma opção mais vantajosa em relação ao SISCOM II, pois sua acurácia é apenas 0,33% menor, enquanto o tempo de execução é quase a metade do tempo necessário para executar o SISCOM II. Nesse contexto, o ganho marginal na acurácia do SISCOM II não se justifica em comparação com o custo computacional para sua execução.

Portanto, o SISCOM I se destaca como a escolha preferencial, uma vez que oferece um desempenho praticamente equivalente em termos de acurácia, mas com uma execução significativamente mais rápida, tornando-o mais eficiente em termos de recursos computacionais. Essa análise reforça a importância de considerar não apenas a acurácia isoladamente, mas também o tempo de execução ao selecionar o classificador mais adequado para uma determinada aplicação.

Capítulo 7

Conclusão e Trabalhos Futuros

Nesta dissertação foi abordada a tarefa reconhecimento e classificação de emoções humanas básicas a partir de imagens digitais através do contexto de aprendizado de máquina, aprendizado profundo. Foram utilizadas técnicas existentes na literatura como as redes de aprendizado profundo CNN.

Todos os testes realizados nessa dissertação utilizaram o dataset consolidado desenvolvido para essa pesquisa a partir de quatro bases de dados: RaFD databases Radboud Faces, CK+ Extended Cohn-Kanade, IMPA-FACE3D e FACES.

Visando determinar a melhor arquitetura dos modelos de Classificadores Individuais, foi realizado um processo de finetunning a partir de uma arquitetura base e definido variações dos metaparâmetros dessa rede com objetivo de testar diferentes configurações de arquitetura e analisar os melhores desempenhos. Para analisar os resultados, foi utilizado um conjunto de métricas de desempenho e testes estatísticos, que verificaram os desempenhos de todos os modelos de Classificadores Individuais.

Foram realizadas três fases de experimentos, a primeira analisando os metaparâmetros Optmizer function e Learning Rate que foi possível observar os melhores resultados para as configurações utilizando o algoritmo de ADAM combinado com uma Learaning Rate igual a 0.0001. Já para a segunda fase de testes foi analisado o metaparâmetro de Neurons da rede DNN, determinou-se o melhor resultado para uma configuração da rede DNN com um número de Neurons igual a 256. Para a terceira e última fase foi analisado o metaparâmetro de número de Convolutional layers da rede CNN, observou-se o melhor resultado com uma configuração utilizando 5 camadas de convolução para rede CNN.

Assim, com os resultados apresentados os seguintes objetivos específicos dessa dissertação podem ser esclarecidos: *"Realizar o processo de "finetuning" da rede proposta utilizando algoritmos de CNNs para determinar a melhor arquitetura e metaparâmetros para a tarefa de classificação de emoções humanas básicas a partir de imagens digitiais"*.

Com base nos resultados obtidos do processo de finetunning realizado, podemos concluir que a melhor arquitetura para a tarefa de classificação de emoções humanas básicas a partir de imagens

digitiais é uma rede com a seguinte configuração: *CNN(5) - DNN(256,256,7) - ADAM(0.0001)*. Por tanto, uma rede utilizando um CNN com cinco camadas de Convolutional layers associado uma rede DNN com três camadas de Dense Layers utilizando o número de Neurons igual 256 e Optimizer function como ADAM em combinação com uma Learning Rate igual 0.0001.

A segunda etapa dessa dissertação foi realizar o desenvolvimento de um Sistema Comitê de classificadores, essa solução nomeada de SISCOM, visando a melhoria do desempenho dos modelos únicos. A metodologia aplicada foi de selecionar 5 modelos de Classificadores Individuais que obtiveram os melhores resultados de performance durante o processo de finetunning para composição do pool de classificadores, além disso seriam desenvolvidas duas configurações para os sistemas de comitês de classificadores com objetivo de averiguar a influência do número de classificadores individuais no pool de classificadores. Assim, foram desenvolvidos o SISCOM I com uma configuração utilizando 3 modelos de Classificadores Individuais e SISCOM II com uma configuração utilizando 5 modelos de Classificadores Individuais.

Assim, com os resultados apresentados os seguintes objetivos específicos dessa dissertação podem ser esclarecidos: *"Determinar o melhor número de Classificadores Individuais integrados para composição do Sistema Comitê"*.

Com base nos resultados obtidos das duas implementações, observou-se um desempenho excelente da solução SISCOM nas métricas analisadas atingindo uma acurácia igual a 95,569% e um custo computacional baixo comparado aos modelos de Classificadores Individuais. Em comparação as duas configurações SISCOM I e II, os resultados de acurácia ficaram extremamente próximos, mas o custo computacional aumentou na configuração de 5 modelos de Classificadores Individuais. Assim, é possível concluir que não houve ganhos de desempenhos suficientes com o aumento do número de Classificadores Individuais no denominado pool de classificadores que justificam o aumento do custo computacional.

Em relação ao último objetivo específico estabelecido dessa dissertação: *"Analisar qual a melhor solução em relação as métricas de performance e estatísticas geradas, além do custo computacional para o processo de reconhecimento e classificação de emoções humanas básicas a partir de imagens digitais"*.

Com base nos resultados gerais obtidos ao longo da pesquisa, conclui-se que a técnica de Sistemas de Múltiplos Classificadores é extremamente eficiente no seu objetivo de melhorar a precisão e a generalização de um modelo de aprendizado de máquina. Associado a técnica de finetunning realizada para os modelos Classificadores Individuais foi possível obter a arquitetura e as configurações dos metaparâmetros da rede proposta para o melhor desempenho em relação ao objetivo da dissertação de reconhecimento e classificação de emoções humanas a partir de imagens digitais.

Em conclusão, a melhor solução para problemática é a utilização da técnica de Sistemas de Múltiplos Classificadores. Em específico a solução SISCOM I, foi a implementação com melhor resultado em performance das métricas de avaliação com uma acurácia de 95,248% e um custo computacional de 27 minutos.

Como trabalhos futuros dessa dissertação podem ser citados: *(i)* a criação de um banco de dados mais extenso com uma gama mais ampla de imagens digitais; *(ii)* a implementação de um sistema de reconhecimento de emoções baseado em dados de voz; *(iii)* a implementação de um sistema de reconhecimento de informações baseado em vídeo em tempo real para reconhecimento de emoções de usuários.

Referências

1 AGARWAL MAYANK; AGRAWAL, Himanshu et al. Face Recognition Using Principle Component Analysis, Eigenface and Neural Network. **IEEE**, 2010. Disponível em: <https://ieeexplore.ieee.org/document/5432754>.

2 AMIDI, Prof. Shervine Amidi Prof. Afshine. **Convolutional Neural Networks Cheatsheet**. [S.l.: s.n.]. Disponível em: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>.

3 ANDREJ KARPATHY, Justin Johnson; LI, Fei-Fei. **Convolutional Neural Networks (CNNs / ConvNets)**. [S.l.: s.n.]. Disponível em: <https://cs231n.github.io/convolutional-networks/>.

4 ARAÚJO, DÉBORA DA CONCEIÇÃO. **AVALIAÇÃO DE COMITÊS COM CLASSIFICADORES TRADICIONAIS E PROFUNDOS PARA ANÁLISE DE SENTIMENTOS**. 2019. F. 112. Master's thesis – Universidade Federal de Pernambuco. Disponível em: <https://repositorio.ufpe.br/handle/123456789/33691>.

5 BACKES, Andre; JUNIOR, Jarbas. In: INTRODUÇÃO à Visão Computacional Usando MATLAB. [S.l.]: Altas Books, 2016.

6 BOHMRAH, Maneet Kaur Harjot. Classification of Covid-19 patients using efficient fine-tuned deep learning DenseNet model. **KeAI**, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666285X21000315>.

7 COSTA VIEIRA, Prof. Dr. Marcelo Andrade da. **VISÃO COMPUTACIONAL**. [S.l.: s.n.], jul. 2020. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/5549058/mod_resource/content/1/Aula%201%20%20Introducao%20a%20Visao%20Computacional.pdf>.

8 CRNKOVIC, Prof.a. Dr.a Luciana Helena. **Inteligência emocional**. [S.l.: s.n.], jan. 2020. Disponível em: <https://sistemas-biblioteca.eesc.usp.br/semana_pos2020/doc/inteligencia.pdf>.

9 DAMÁSIO, António. In: O ERRO DE DESCARTES Emoção, razão e o cérebro humano. [S.l.: s.n.], 2012.

10 DARWIN, Charles. In: THE Expression of the Emotions in Man and Animals. [S.l.: s.n.], 1897.

11 DIETTERICH THOMAS BAKIRI, Ghulum. Solving Multiclass Learning Problems via Error-Correcting Output Codes. **Journal of artificial intelligence research**, 1994.

12 DINIZ, FÁBIO ABRANTES. RedFace – Um Sistema de Reconhecimento de Expressões Faciais para Apoiar um Ambiente Virtual de Aprendizagem. **UERN**, 2013.

13 DIOGO, Paula. Sobre as Emoções Humanas e o Cuidar de Enfermagem. **Escola Superior de Enfermagem de Lisboa**, 2020. Disponível em: <https://www.researchgate.net/publication/338966290_Editorial_Revista_Pensar_Enfermagem_Sobre_as_Emocoes_Humanas_e_o_Cuidar_de_Enfermagem>.

14 EDUARDO CARVALHO GERALDO P. ROCHA FILHO, Vinícius P. Gonçalves et al. Exploiting Convolutional Neural Networks to Recognize User's Emotion. **UnB**, v. 8, 2020.

15 EKMAN, Paul. In: THE Expression of the Emotions in Man and Animals. [S.l.: s.n.], 2006.

16 FILHO LUIZ CAVALCANTI, George. Uma arquitetura para combinac ao de classificadores otimizadapor m etodos de poda com aplicac ao emcredit scoring. **UFPE**, 2014.

17 FLORINDO, Jo˜ao B. **Redes Neurais Convolucionais - Deep Learning**. [S.l.: s.n.], jan. 2018. Disponível em: <https://www.ime.unicamp.br/˜jbflorindo/Teaching/2018/MT530/T10.pdf>.

18 GONZALEZ, Woods. In: DIGITAL Image Processing. [S.l.: s.n.], 2018.

19 IAN GOODFELLOW, Yoshua Bengio; COURVILLE, Aaron. Deep Learning. Book in preparation for MIT Press. [S.l.], 2016. Disponível em: <http://www.deeplearningbook.org>.

20 J. MENA-CHALCO, R. Cesar-Jr; VELHO, L. Banco de dados de faces 3d: Impa-face3d. **Tech. Rep**, 2008.

21 JAIN, Stan Z. Li Anil K. In: HANDBOOK of Face Recognition. [S.l.]: Springer Publishing Company, 2011.

22 LEANDRO Y. MANO GERALDO P. ROCHA FILHO, Jó Ueyama et al. A Low-Cost Smart Home Automation to Enhance Decision-Making based on Fog Computing and Computational Intelligence. **IEE**, 2018.

23 LECUN YANN; BENGIO, Yoshua et al. Deep Learning. **Nature Publishing Group**, 2015. Disponível em: <https://www.cs.toronto.edu/˜hinton/absps/NatureDeepReview.pdf>.

24 LI, Prof. Fei-Fei. **Lecture 5: Convolutional Neural Networks**. [S.l.: s.n.]. Disponível em: <http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf>.

25 MENDONÇA, THOMAS DILLAN BALTAZAR. Sistema de Reconhecimento de Expressões Faciais para Classi- ficação de Emoções de Usuários em Sistemas Computacionais. **UFC**, 2018.

26 MURPHY, Kevin. In: MACHINE Learning: A Probabilistic Perspective. [S.l.]: The MIT Press, 2012.

27 N. C. EBNER, M. Riediger; LINDENBERGER, U. Faces—a database of facial expressions in young, middle-aged, and older women and men: Development and validation. **Behavior research methods**, 2010.

28 NANGIR, Cagatay Catal Mehmet. A sentiment classification model based on multiple classifiers. **Elsevier**, 2017.

29 O. LANGNER R. DOTSCH, G. Bijlstra et al. Presentation and validation of the radboud faces database. **Cognition and Emotion**, 2010.

30 OPAS. **OMS destaca necessidade urgente de transformar saúde mental e atenção**. [S.l.: s.n.], jun. 2022. Disponível em: <https://www.paho.org/pt/noticias/17-6-2022-oms-destaca-necessidade-urgente-transformar-saude-mental-e-atencao>.

31 P.LUCEY J.F.COHN, T.Kanade et al. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. **IEE**, 2010.

32 ROLI; FABIO; GIACINTO GIORGIO VERNAZZA, Gianni. Methods for Designing Multiple Classifier Systems. **International Workshop on Multiple Classifier Systems**, 2001.

33 TOO ENDA; YUJIAN, Li et al. A comparative study of fine-tuning deep learning models for plant disease identification. **Elsevier**, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168169917313303?casa_token=ZyGcX28Q9V0AAAAA:NjxQMftJGLLfAHKQ5m089IYN_KX6ZlcAqXFso2PYpf043gTRH5I-OJTtfIYnIGiiUVPoYZs>.